Architecture of ML Systems (AMLS)
08 Hardware Accelerators

Prof. Dr. Matthias Boehm
Technische Universität Berlin
Berlin Institute for the Foundations of Learning and Data
Big Data Engineering (DAMS Lab)

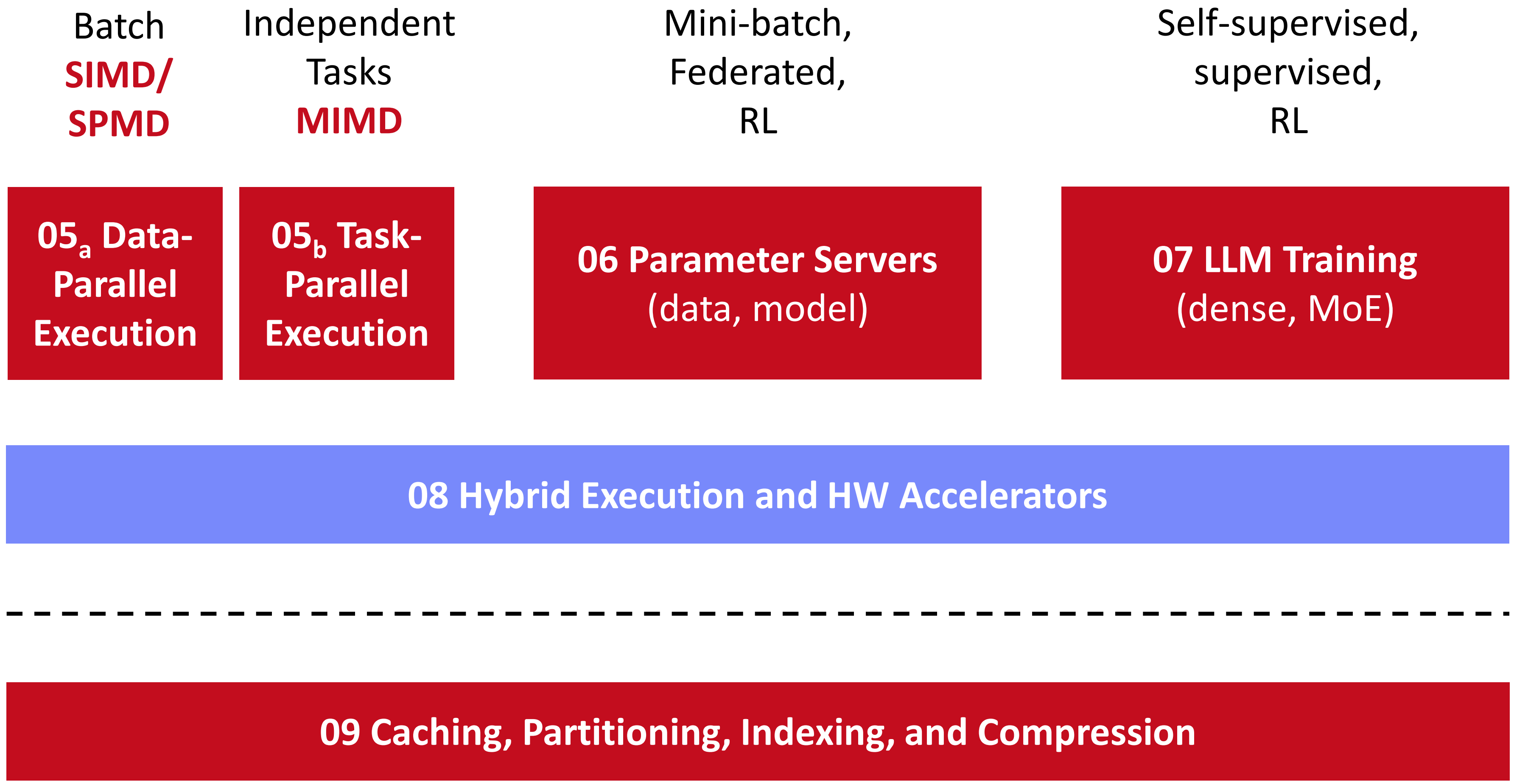
Last update: Jun 11, 2026

Announcements / Org

- **#1 Hybrid & Video Recording**
  - Hybrid lectures (in-person, zoom) with optional attendance
    https://tu-berlin.zoom.us/j/9529634787?pwd=R1ZsN1M3SC9BOU1OcFdmem9zT202UT09
  - Zoom **video recordings**, links from website
    https://mboehm7.github.io/teaching/ss26_amls/index.htm

- **#2 Exam Registration**
  - Thu **July 23, 4-6pm** (room TBD) → 5 registrations
  - Thu **Aug 06, 4-6pm** (room TBD) → 5 registrations
  - Thu **Aug 27, 4-6pm** (room TBD) → 1 registrations

Categories of Execution Strategies



Batch
SIMD/ SPMD

Independent Tasks
MIMD

Mini-batch, Federated, RL

Self-supervised, supervised, RL

05_a Data-Parallel Execution

05_b Task-Parallel Execution

06 Parameter Servers
(data, model)

07 LLM Training
(dense, MoE)

08 Hybrid Execution and HW Accelerators

09 Caching, Partitioning, Indexing, and Compression

Agenda

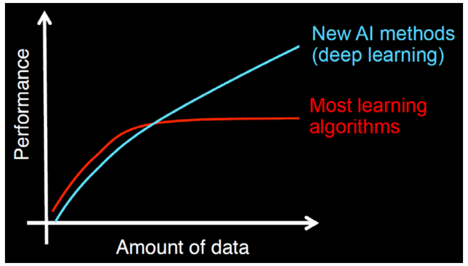
- **Motivation and Terminology**
- **GPUs in ML Systems**
- **FPGAs in ML Systems**
- **ASICs and other HW Accelerators**

Motivation and Terminology

Recap: Driving Factors for ML

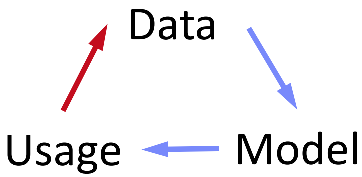
- **Improved Algorithms and Models**
  - Success across data and application domains (e.g., health care, finance, transport, production)
  - More complex models which leverage large data

[**Credit:** Andrew Ng'14]



- **Availability of Large Data Collections**
  - Increasing automation and monitoring ➔ data (simplified by cloud computing & services, annotation services)
  - Feedback loops, **simulation/data prog./augmentation**
    → Trend: **self-supervised learning** (*-GPT-x)

Feedback Loop



- **HW & SW Advancements**
  - Higher performance of hardware and infrastructure (cloud)
  - Open-source large-scale computation frameworks, ML systems, and vendor-provides libraries

DNN Challenges

- **#1 Larger Models and Scoring Time**

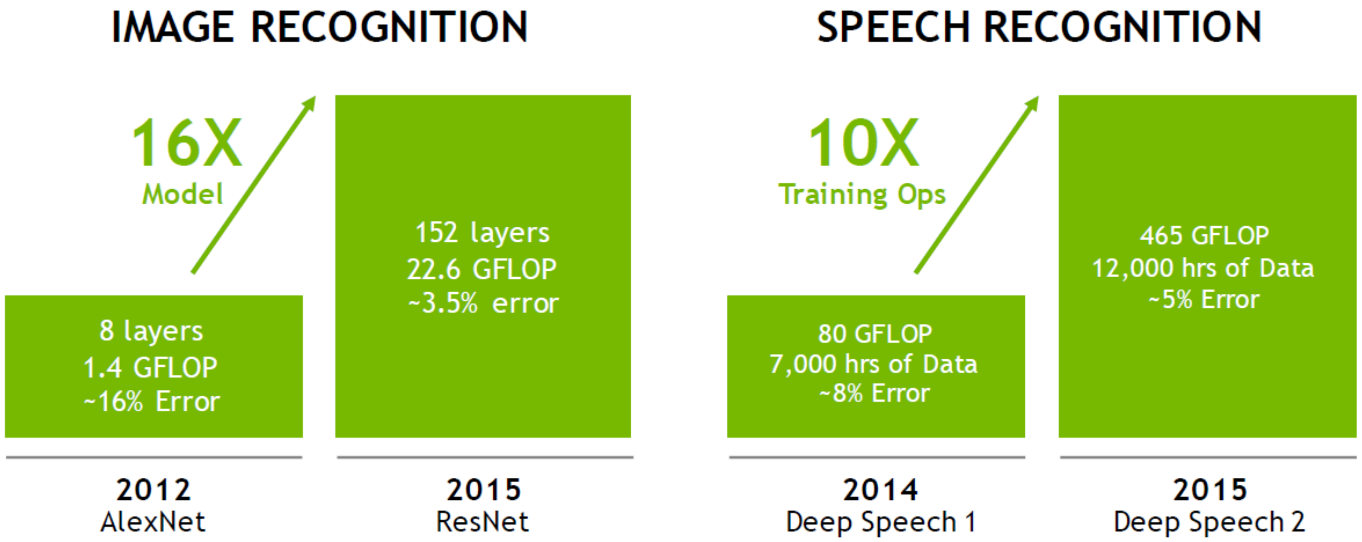


IMAGE RECOGNITION

16X Model

8 layers
1.4 GFLOP
~16% Error

2012
AlexNet

152 layers
22.6 GFLOP
~3.5% error

2015
ResNet

SPEECH RECOGNITION

10X Training Ops

80 GFLOP
7,000 hrs of Data
~8% Error

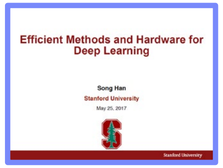
2014
Deep Speech 1

465 GFLOP
12,000 hrs of Data
~5% Error

2015
Deep Speech 2

- **#2 Training Time**
  - **ResNet18:** 10.76% error, 2.5 days training
  - **ResNet50:** 7.02% error, 5 days training
  - **ResNet101:** 6.21% error, 1 week training
  - **ResNet152:** 6.16% error, **1.5 weeks training**
- **#3 Energy Efficiency**

But, LLM Scaling Laws (see 08)



Efficient Methods and Hardware for Deep Learning

Song Han
Stanford University
May 25, 2017

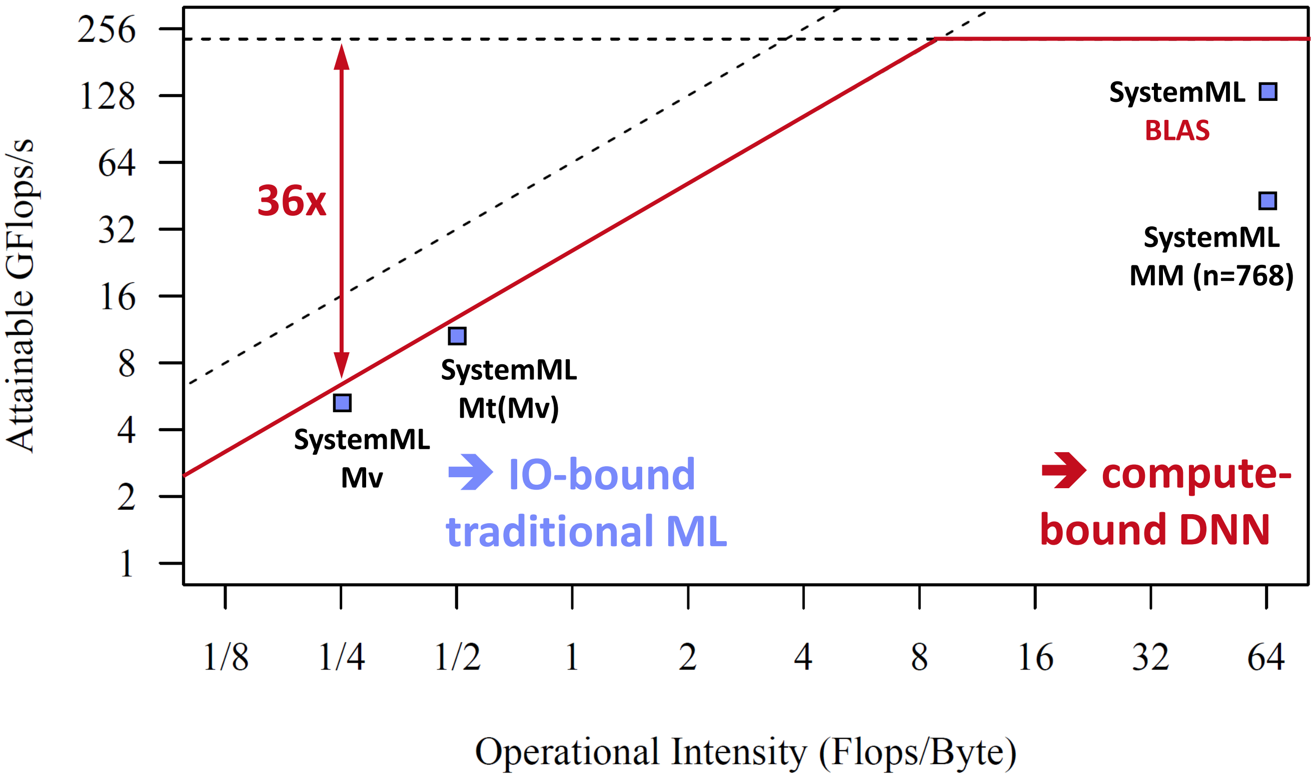
Stanford University

[Song Han: Efficient Methods and Hardware for Deep Learning, Stanford cs231n, 2017]

Excursus: Roofline Analysis

- **Setup:** 2x6 E5-2440 @2.4GHz–2.9GHz, DDR3 RAM @1.3GHz (ECC)
  - Max mem bandwidth (local): 2 sock x 3 chan x 8B x 1.3G trans/s → **2 x 32GB/s**
  - Max mem bandwidth (QPI, full duplex) → **2 x 12.8GB/s**
  - Max floating point ops: 12 cores x 2*4dFP-units x 2.4GHz → **2 x 115.2GFlops/**
- **Roofline Analysis**
  - Off-chip memory traffic
  - Peak compute

[S. Williams, A. Waterman, D. A. Patterson: Roofline: An Insightful Visual Performance Model for Multicore Architectures. **Commun. ACM 2009**]



(Experiments from 2017)

HW Challenges

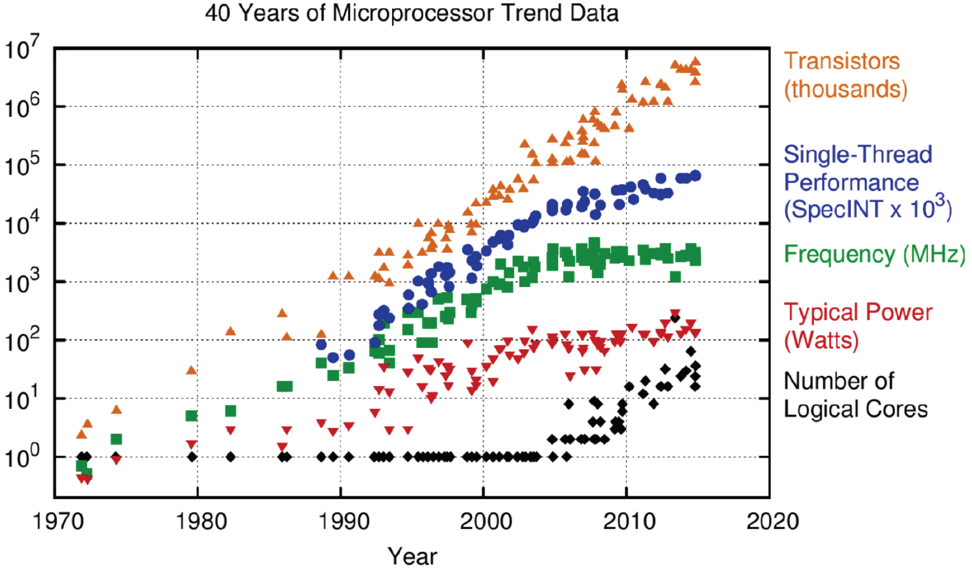
[S. Markidis, E. Laure, N. Jansson, S. Rivas-Gomez and S. W. D. Chien: Moore's Law and Dennard Scaling]


Moore's Law and Dennard Scaling

- **#1 End of Dennard Scaling** (~2005)
  - **Law:** power stays proportional to the area of the transistor
  - Ignored leakage current / threshold voltage
    → **increasing power density S^2** (power wall, heat) → stagnating frequency

$P = \alpha\ CFV^2$ (**power density 1**)
(P .. Power, C .. Capacitance,
F .. Frequency, V .. Voltage)

- **#2 End of Moore's Law** (~2010-20)
  - **Law:** #transistors/performance/ CPU frequency doubles every 18/24 months
  - Original: # transistors per chip doubles every two years **at constant costs**
  - Now increasing costs (10/7/5nm)

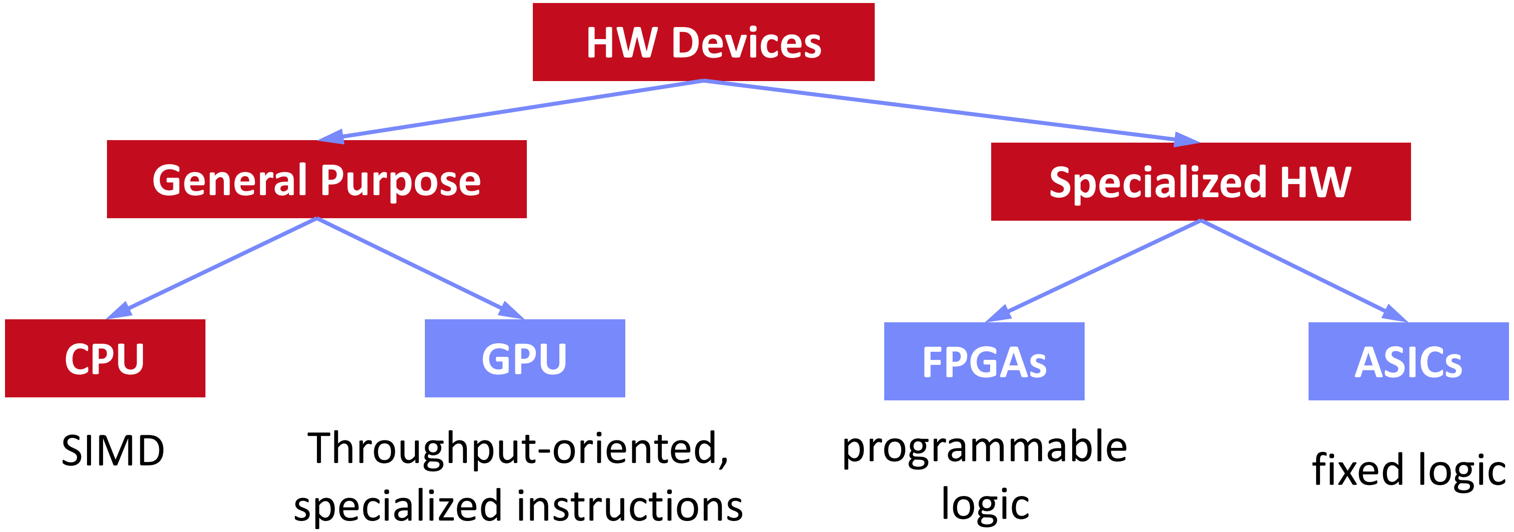


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

➔ **Consequences: Dark Silicon and Specialization**

Towards Specialized Hardware

- **HW Specialization**


HW Devices

General Purpose

Specialized HW

CPU

SIMD

GPU

Throughput-oriented, specialized instructions

FPGAs

programmable logic

ASICs

fixed logic

- **Additional Specialization**
  - **Data Transfer & Types:** e.g., low-precision, quantization
  - **Sparsity Exploitation:** e.g., sparsification, exploit across ops, defer weight decompression just before instruction execution
  - **Near-Data Processing:** e.g., operations in main memory, storage class memory (SCM), secondary storage (e.g., SSDs), and tertiary storage (e.g., tapes)

08 Caching, Indexing and Compression

GPUs in ML Systems

NVIDIA Volta V100 – Specifications

- **Tesla V100 NVLink**
    - FP64: 7.8 TFLOPs, FP32: 15.7 TFLOPs
    - DL FP16: 125 TFLOPs
    - NVLink: 300GB/s
    - Device HBM: 32 GB (900 GB/s)
    - Power: 300 W
- **Tesla V100 PCIe**
    - FP64: 7 TFLOPs, FP32: 14 TFLOPs
    - DL FP16: 112 TFLOPs
    - PCIe: 32 GB/s
    - Device HBM: 16 GB (900 GB/s)
    - Power: 250 W

[Credit: https://nvidia.com/de-de/data-center/tesla-v100/]

NVIDIA Volta V100 – Architecture

[NVIDIA Tesla V100 GPU Architecture - THE WORLD'S MOST ADVANCED DATA CENTER GPU, Whitepaper, **Aug 2017**]

- **6 GPU Processing Clusters (GPCs)**
  - 7 Texture Processing Clusters (TPC)
  - 14 Streaming Multiprocessors (SM)

NVIDIA Volta V100 – SM Architecture

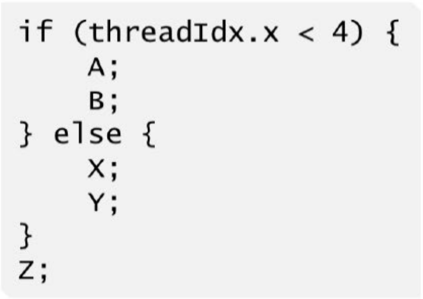
- **FP64 cores: 32 / FP32 cores: 64**
- **INT32 cores: 64**
- **“Tensor cores”: 8**
- **Max warps /SM: 64**
- **Threads/warp: 32**

Single Instruction Multiple Threads (SIMT)

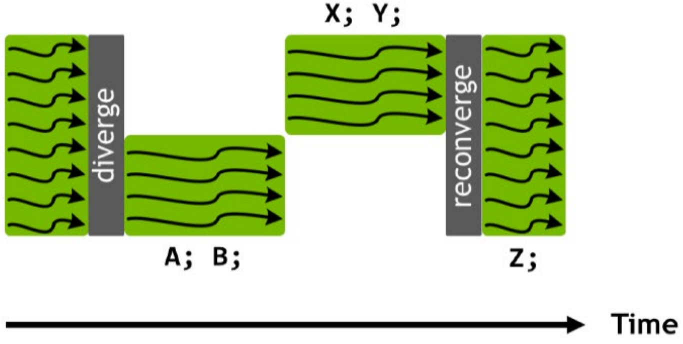
- **32 Threads grouped to warps and execute in SIMT model**

Thread Divergence

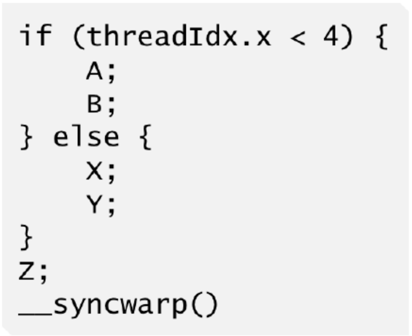
- **Pascal P100 Execution Model**
  - Warps use a single program counter + active mask



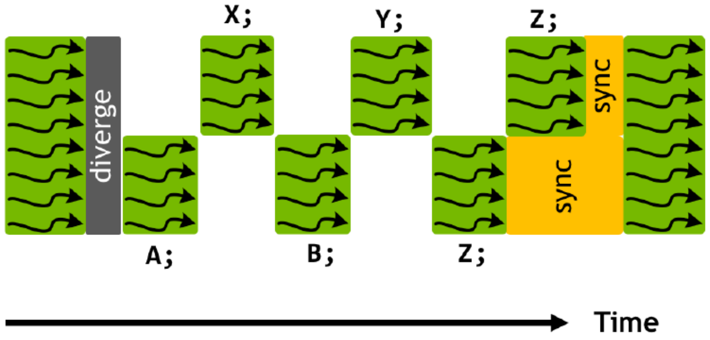
```
if (threadIdx.x < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```



- **Volta V100 Execution Model**
  - Independent thread scheduling
  - Per-thread program counters and call stacks
  - New `__syncwarp()` primitive (if needed) + convergence optimizer



```
if (threadIdx.x < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
__syncwarp()
```



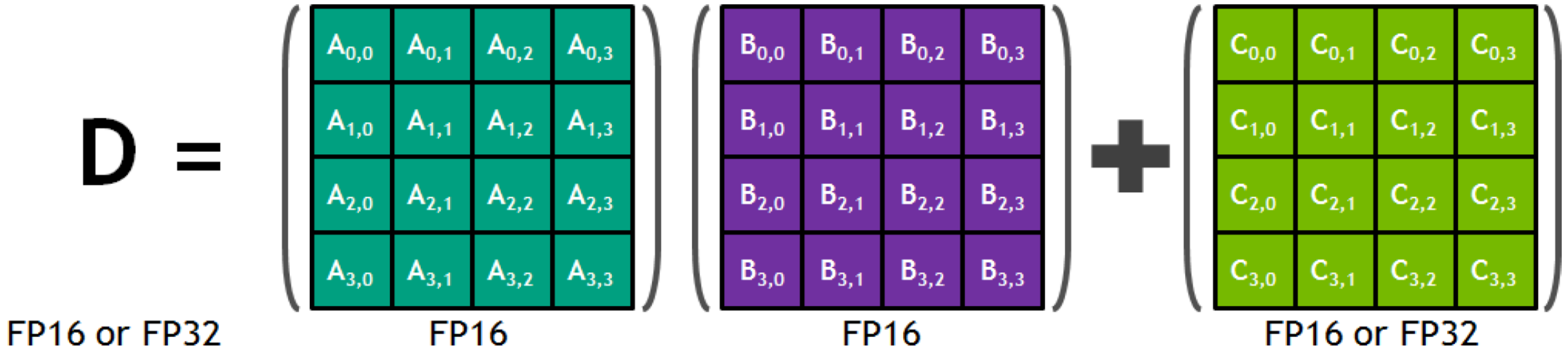
NVIDIA Volta V100 – Tensor Cores

- **“Tensor Core”**
  - Specialized instruction for **4x4 by 4x4 fused matrix multiply**
  - Two FP16 inputs and FP32 accumulator
  - Exposed as warp-level matrix operations w/ special load, mm, acc, and store

[Bill Dally: Hardware for Deep Learning. **SysML 2018**]

D = A %*% B + C

64 FMA operations

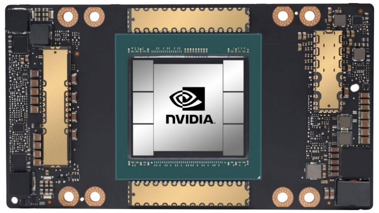


$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0}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

NVIDIA Ampere A100

- **Specification**
  - 7nm, 8 GPC x 8 TPC * 2 SM = 128 SMs, 40GB HBM
  - FP64: 9.7 TFLOPs / FP64 TensorCore: 19.5 TFLOPs
  - FP32 **19.5 TFLOPs**, FP16: 78 TFLOPs, BF16: 39 TFLOPs
  - TF32 TensorCore 156 TFLOPs / 312 TFLOPs (sparse)
  - FP16 TensorCore 312 TFLOPs / 624 TFLOPs (sparse), INT8, INT4

[NVIDIA A100 Tensor Core GPU Architecture - UNPRECEDENTED ACCELERATION AT EVERY SCALE, Whitepaper, **Aug 2020**]



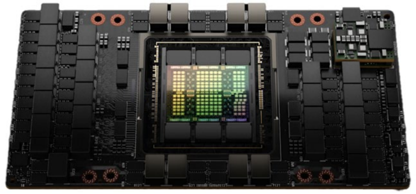
- **New Features**
  - New generation of “TensorCores” (FP64, **new data types:** TF32, BF16)
  - Fine-grained **sparsity exploitation**
  - Multi-instance **GPU (MIG) virtualization**: up to 7 virtual GPU instances
  - Link technologies: **NVLink 3** (25GB/s bidirectional) x 12 links = 600GB/s
  - **Submission of task graphs** (launch a workflow of kernels)

NVIDIA Hopper H100, GH200

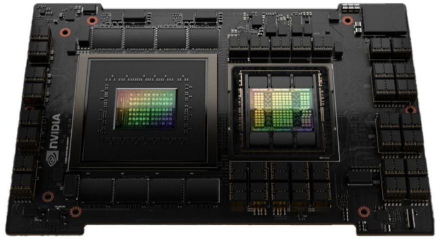
- **Specification** SXM5 / PCIe
  - 7nm, 7/8 GPC x 9 TPC * 2 SM = 114/144 SMs, 80GB HBM
  - FP64: 25.6 TFLOPs / FP64 TensorCore: 66.9 TFLOPs
  - FP32 **66.9 TFLOPs**, FP16: 134 TFLOPs, BF16: 134 TFLOPs
  - TF32 TensorCore 495 TFLOPs / 989 TFLOPs (sparse)
  - FP16 TensorCore 989 TFLOPs / 1979 TFLOPs (sparse),
  - **FP8 TensorCore** 1979 TFLOPs / 3958 TFLOPs (sparse), INT8
- **New Features**
  - **Dedicated Transformer Engine** (hybrid FP8 and FP16)
  - HBM3 memory and 50MB L2 cache; **TMA - tensor mem. acc.** (async copy)
  - 2nd Gen Multi-instance **GPU (MIG) virtualization**: up to 7 virtual GPUs
  - Confidential computing (**trusted execution environments**)
  - Improved link technologies (NVLink 4, NVSwitch 3, PCIe 5)
- **NVIDIA Grace Hopper DGX GH200**
  - 256 H100 GPUs in 16 Racks, 96L1 + 36L2 NVSwitches

[NVIDIA H100 Tensor Core GPU Architecture
EXCEPTIONAL PERFORMANCE, SCALABILITY,
AND SECURITY FOR THE DATA CENTER,
Whitepaper, **May 2023**]

H100



GH200
(GPU+
ARM CPU)

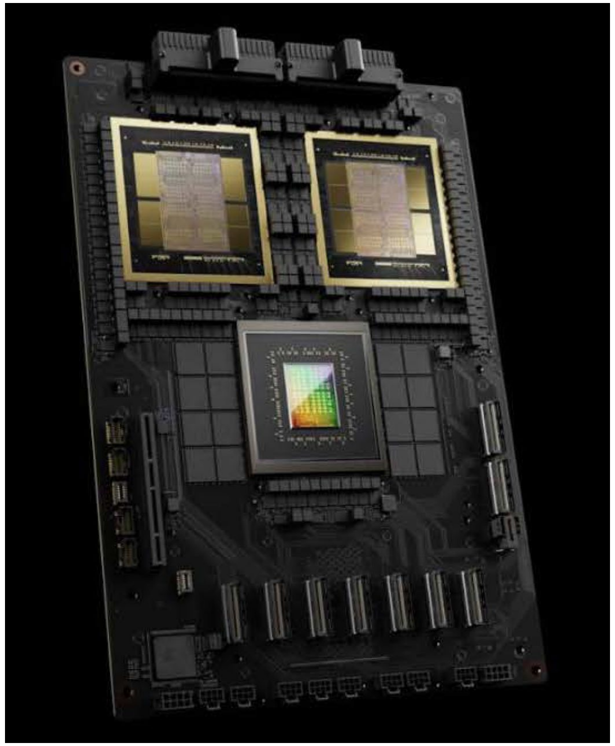


[https://www.hpcwire.com/2023/05/28/nvidia-announces-new-1-exaflops-ai-supercomputer-grace-hopper-in-full-production/]

NVIDIA Blackwell B100, B200, GB200

[NVIDIA Blackwell Architecture Technical Brief: Powering the New Era of Generative AI and Accelerated Computing, Whitepaper, **04/2024**]

- **New Features**
  - B100/B200 chips with **2 dies**, **cache-coherence**, and **10 TB/s** chip-to-chip high-bandwidth interface (NV-HBI)
  - 2nd Gen **Transformer Engine** (hybrid FP8 and FP16)
  - **FP4 TensorCore** + UINT4 quantization
  - **NVLink v5** (2x900GB/s) and **NVLink switch chip** (130TB/s bandwidth)
- **GB200 Superchip**
  - **1 Grace CPU and 2 Blackwell GPUs** (see right)
  - **FP64:** 90 TFLOPs; **FP32:** 180 TFLOP
  - **TF32 dense/sparse:** 2.5/5 PFLOPs; **BF16:** 5/10 PFLOPs
  - **FP8 dense/sparse:** 10/20 PFLOPs; **FP4:** 20/40 PFLOPs
- **NVL72**
  - **36 GB200 superchips in one rack** (72 GPUs, see right)
  - **FP4 dense/sparse:** 0.7/1.4 EFLOPs



Excursus: Amdahl's Law

- **Amdahl's law**
  - Given a fixed problem size, **Amdahl's law gives the maximum speedup**
  - T is the execution time, **s is the serial fraction**, and p the number of processors

Execution Time $$T_p = \frac{(1-s)T}{p} + sT$$

Speedup $$S_p = \frac{T}{T_p}$$

→

Upper-Bound Speedup $$\overline{S_p} = \lim_{p \to \infty} S_p = \frac{1}{s}$$

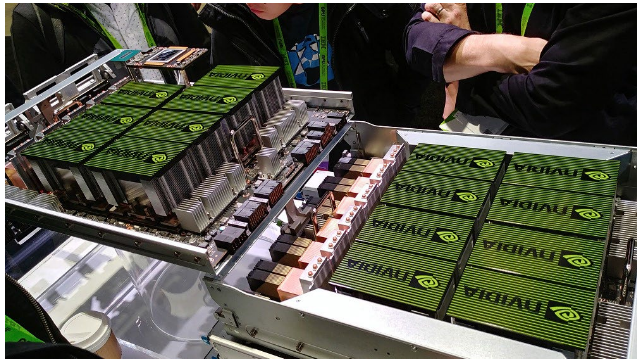
- **Examples**
  - Serial fraction s = 0.01 → max S_p = 100
  - Serial fraction s = **0.05** → max S_p = **20**
  - Serial fraction s = **0.1** → max S_p = **10**
  - Serial fraction s = 0.5 → max S_p = 2

GPUs for DNN Training

- **GPUs for DNN Training** (2009)
  - Deep belief networks
  - Sparse coding

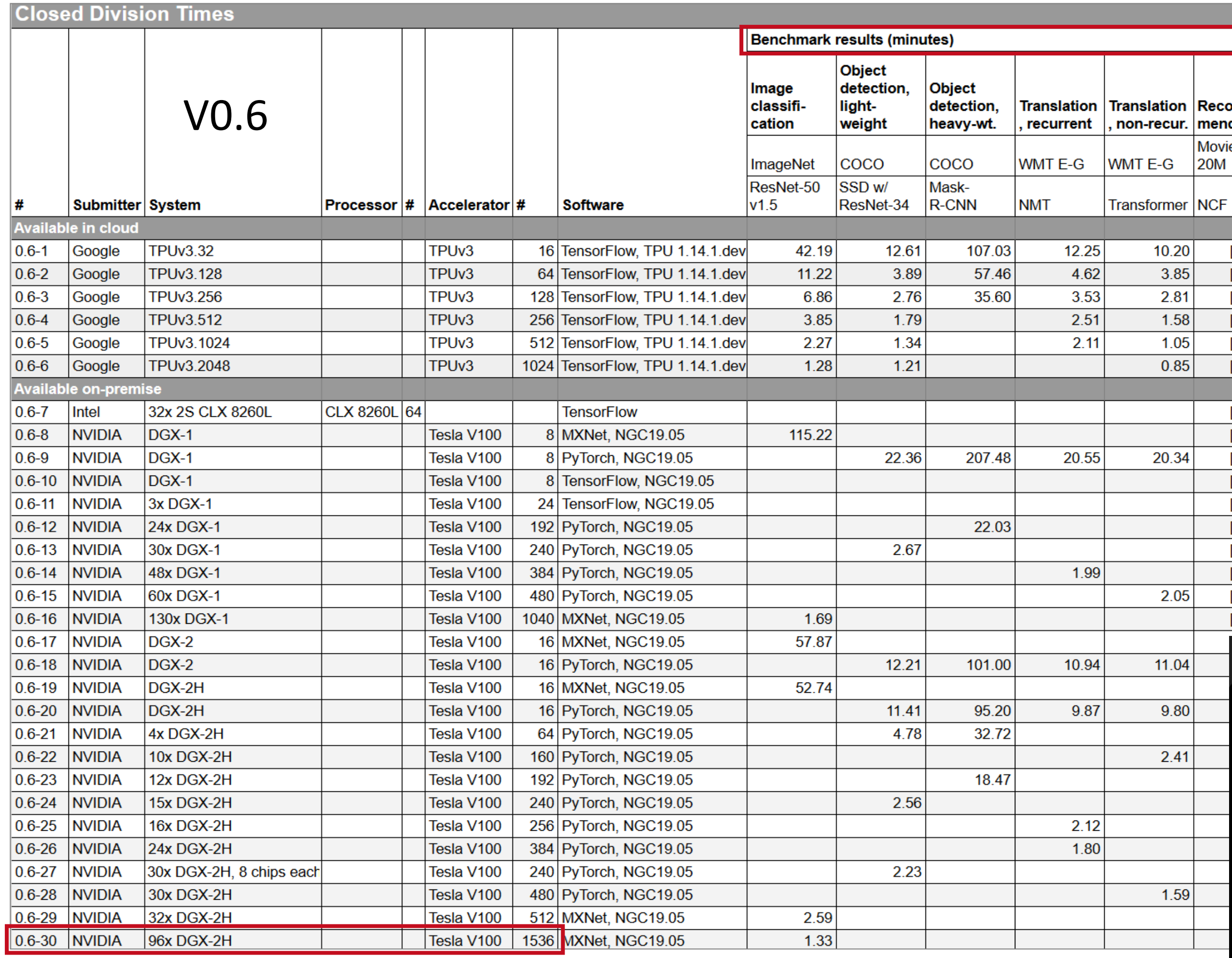
[Rajat Raina, Anand Madhavan, Andrew Y. Ng: Large-scale deep unsupervised learning using graphics processors. **ICML 2009**]

- **Multi-GPU Learning** (Now)
  - Exploit multiple GPUs with a mix of **data- and model-parallel parameter servers**
  - Dedicated ML systems for multi-GPU learning
  - Dedicated HW: e.g., NVIDIA DGX-1 (8xP100), **NVIDIA DGX-2 (16xV100, NVSwitch)**, NVIDIA DGX A100 (8x A100, NVSwitch, Mellanox) NVIDIA DGX H100 (8x H100, Mellanox InfiniBand)



- **DNN Framework support**
  - All specialized DNN frameworks have very good support for GPU training
  - Most of them also support multi-GPU training

Recap: DNN Benchmarks



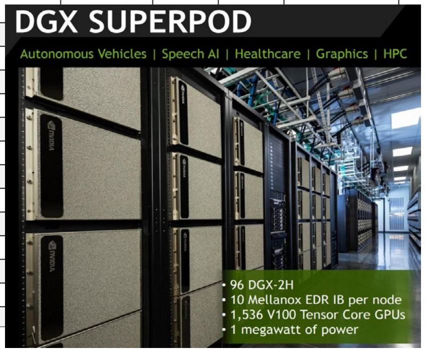
Closed Division Times

V0.6

#	Submitter	System	Processor	#	Accelerator	#	Software	Benchmark results (minutes)							Details	Code	Notes
								Image classification	Object detection, light-weight	Object detection, heavy-wt.	Translation, recurrent	Translation, non-recur.	Recommendation	Reinforcement Learning			
								ImageNet	COCO	COCO	WMT E-G	WMT E-G	MovieLens-20M	Go			
								ResNet-50 v1.5	SSD w/ ResNet-34	Mask-R-CNN	NMT	Transformer	NCF	Mini Go			
Available in cloud																	
0.6-1	Google	TPUv3.32			TPUv3	16	TensorFlow, TPU 1.14.1.dev	42.19	12.61	107.03	12.25	10.20	[1]		details	code	none
0.6-2	Google	TPUv3.128			TPUv3	64	TensorFlow, TPU 1.14.1.dev	11.22	3.89	57.46	4.62	3.85	[1]		details	code	none
0.6-3	Google	TPUv3.256			TPUv3	128	TensorFlow, TPU 1.14.1.dev	6.86	2.76	35.60	3.53	2.81	[1]		details	code	none
0.6-4	Google	TPUv3.512			TPUv3	256	TensorFlow, TPU 1.14.1.dev	3.85	1.79		2.51	1.58	[1]		details	code	none
0.6-5	Google	TPUv3.1024			TPUv3	512	TensorFlow, TPU 1.14.1.dev	2.27	1.34		2.11	1.05	[1]		details	code	none
0.6-6	Google	TPUv3.2048			TPUv3	1024	TensorFlow, TPU 1.14.1.dev	1.28	1.21			0.85	[1]		details	code	none
Available on-premise																	
0.6-7	Intel	32x 2S CLX 8260L	CLX 8260L	64			TensorFlow						[1]	14.43	details	code	none
0.6-8	NVIDIA	DGX-1			Tesla V100	8	MXNet, NGC19.05	115.22					[1]		details	code	none
0.6-9	NVIDIA	DGX-1			Tesla V100	8	PyTorch, NGC19.05		22.36	207.48	20.55	20.34	[1]		details	code	none
0.6-10	NVIDIA	DGX-1			Tesla V100	8	TensorFlow, NGC19.05						[1]	27.39	details	code	none
0.6-11	NVIDIA	3x DGX-1			Tesla V100	24	TensorFlow, NGC19.05						[1]	13.57	details	code	none
0.6-12	NVIDIA	24x DGX-1			Tesla V100	192	PyTorch, NGC19.05			22.03			[1]		details	code	none
0.6-13	NVIDIA	30x DGX-1			Tesla V100	240	PyTorch, NGC19.05		2.67				[1]		details	code	none
0.6-14	NVIDIA	48x DGX-1			Tesla V100	384	PyTorch, NGC19.05				1.99		[1]		details	code	none
0.6-15	NVIDIA	60x DGX-1			Tesla V100	480	PyTorch, NGC19.05					2.05	[1]		details	code	none
0.6-16	NVIDIA	130x DGX-1			Tesla V100	1040	MXNet, NGC19.05	1.69					[1]		details	code	none
0.6-17	NVIDIA	DGX-2			Tesla V100	16	MXNet, NGC19.05	57.87									
0.6-18	NVIDIA	DGX-2			Tesla V100	16	PyTorch, NGC19.05		12.21	101.00	10.94	11.04					
0.6-19	NVIDIA	DGX-2H			Tesla V100	16	MXNet, NGC19.05	52.74									
0.6-20	NVIDIA	DGX-2H			Tesla V100	16	PyTorch, NGC19.05		11.41	95.20	9.87	9.80					
0.6-21	NVIDIA	4x DGX-2H			Tesla V100	64	PyTorch, NGC19.05		4.78	32.72							
0.6-22	NVIDIA	10x DGX-2H			Tesla V100	160	PyTorch, NGC19.05					2.41					
0.6-23	NVIDIA	12x DGX-2H			Tesla V100	192	PyTorch, NGC19.05			18.47							
0.6-24	NVIDIA	15x DGX-2H			Tesla V100	240	PyTorch, NGC19.05		2.56								
0.6-25	NVIDIA	16x DGX-2H			Tesla V100	256	PyTorch, NGC19.05				2.12						
0.6-26	NVIDIA	24x DGX-2H			Tesla V100	384	PyTorch, NGC19.05				1.80						
0.6-27	NVIDIA	30x DGX-2H, 8 chips each			Tesla V100	240	PyTorch, NGC19.05		2.23								
0.6-28	NVIDIA	30x DGX-2H			Tesla V100	480	PyTorch, NGC19.05					1.59					
0.6-29	NVIDIA	32x DGX-2H			Tesla V100	512	MXNet, NGC19.05	2.59									
0.6-30	NVIDIA	96x DGX-2H			Tesla V100	1536	MXNet, NGC19.05	1.33									

MLPerf v0.6: https://mlperf.org/training-results-0-6/, **MLPerf** v0.7: https://mlperf.org/training-results-0-7

… MLPerf v2.1 (11/2022)



96 x DGX-2H = 96 * 16

= 1536 V100 GPUs

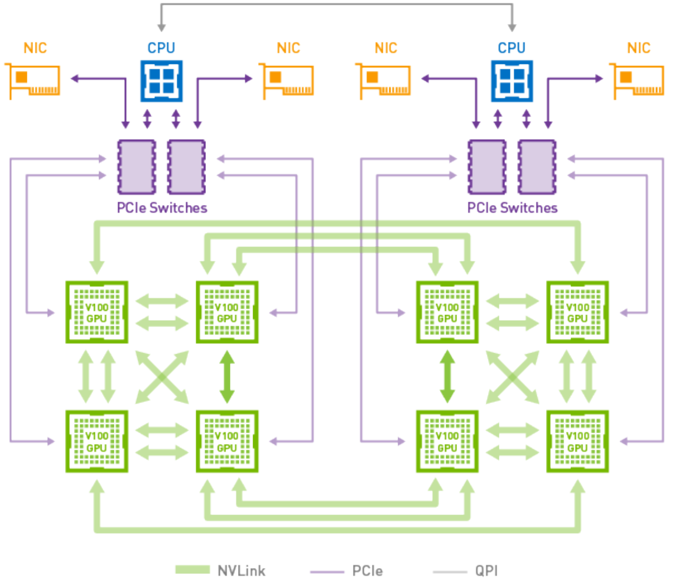
➔ ~ 96 * $400K = **$35M – $40M**

[https://www.forbes.com/sites/tiriasresearch/2019/06/19/nvidia-offers-a-turnkey-supercomputer-the-dgx-superpod/#693400f43ee5]

GPU Link Technologies

- **Classic PCI Express**
  - Peripheral Component Interconnect Express (default)
  - **v3 x16 lanes: 16GB/s**, v4 (2017) x16 lanes: 32GB/s, v5 (2019) x16 lanes: 64GB/s

- **#1 NVLink**
  - Proprietary technology
  - Requires NVLink-enabled CPU (e.g., IBM Power 8/9)
  - Connect GPU-GPU and GPU-CPU
  - NVLink 1: 80+80 GB/s
  - NVLink 2: 150+150 GB/s
  - NVLink 3: 600 GB/s, NVLink 4: 900GB/s, NVLink 5: 2x 900GB/s



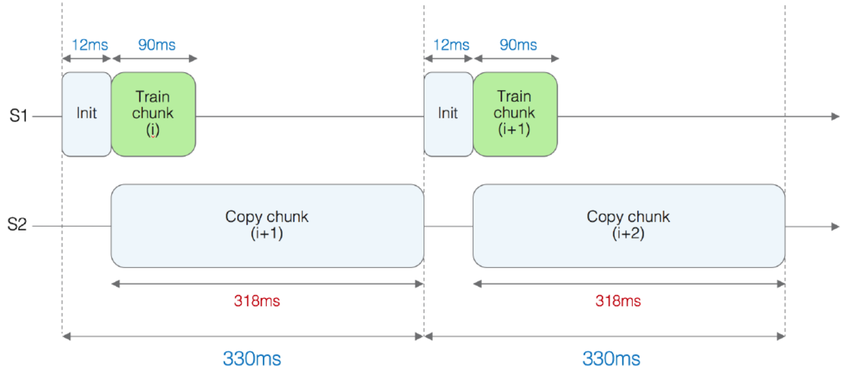
- **#2 NVSwitch**
  - Fully connected GPUs, each communicating at 300GB/s
  - NVSwitch 2 and 3: from 7.2 Tbits/sec to 13.6 Tbits/sec

GPU Link Technologies, cont.

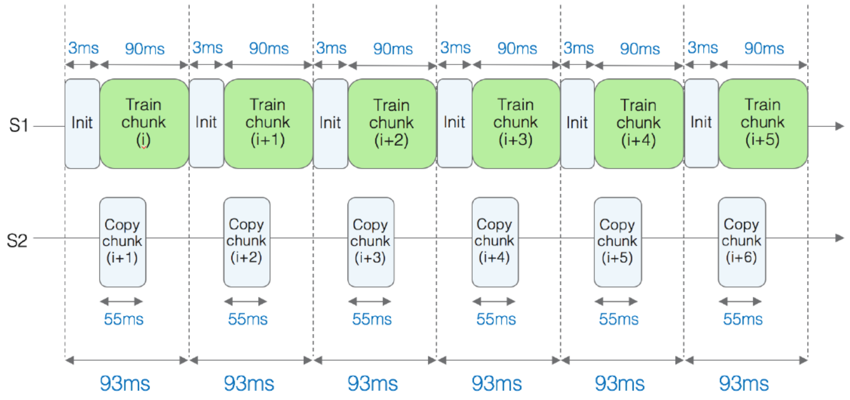
- **Recap: Amdahl's Law**
- **Experimental Setup**
  - **SnapML**, 4 IBM Power x 4 V100 GPUs, NVLink 2.0
  - 200 million training examples of the Criteo dataset (> GPU mem)
  - Train a logistic regression model

[Celestine Dünner et al.: Snap ML: A Hierarchical Framework for Machine Learning. **NeurIPS 2018**]

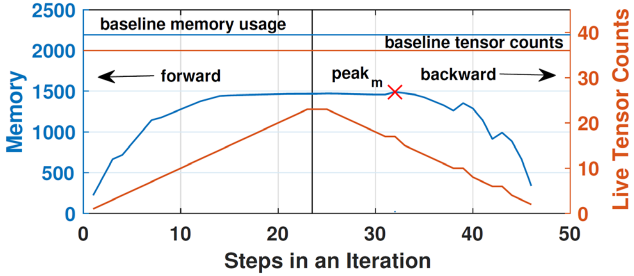
PCIe v3 Interconnect



NVLink Interconnect



Handling Memory Constraints



[Linnan Wang et al: Superneurons: dynamic GPU memory management for training deep neural networks. **PPOPP 2018**]

- **#1 Live Variable Analysis**
  - Remove intermediates ASAP
  - **Examples:** SystemML, TensorFlow, MXNet, Superneurons, MONeT
- **#2 GPU-CPU Eviction**
  - Evict variables from GPU to CPU memory under memory pressure
  - **Examples:** SystemML, Superneurons, GeePS, (TensorFlow)
- **#3 Recomputation**
  - Recompute inexpensive operations (e.g., activations of forward pass)
  - **Examples:** MXNet, Superneurons, MONet, `keras.remat()`
- **#4 Reuse Allocations**
  - Reuse allocated matrices and tensors via free lists, but **fragmentation**
  - **Examples:** SystemML, Superneurons, MONet
- **#5 Physical Operator Selection**
  - Different tradeoffs of performance and size of intermediates (MONet)

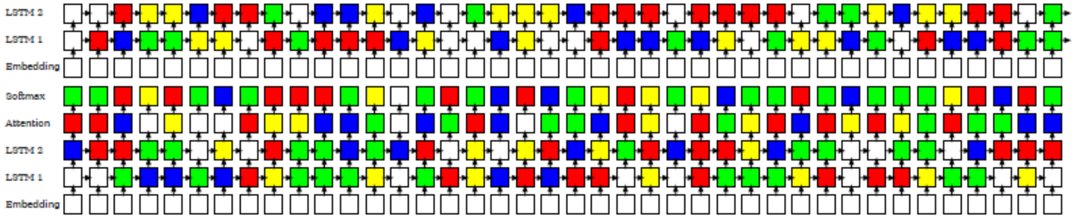
Problem:
Limited
Device Memory

Hybrid CPU/GPU Execution

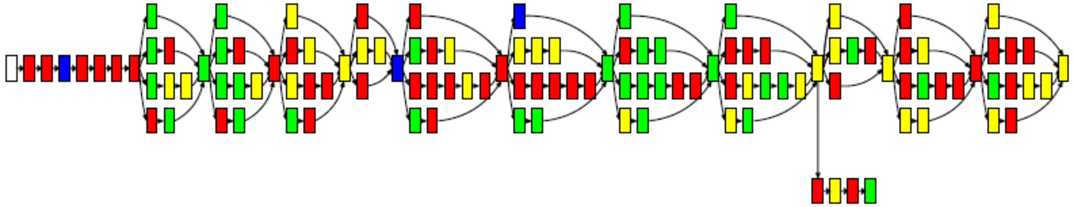
- **Manual Placement**
  - Most DNN frameworks allow manual placement of variables and operations on individual CPU/GPU devices
  - **Heuristics and intuition of human experts**
- **Automatic Placement**
  - Sequence-to-sequence model to predict which operations should run on which device
  - **Examples:**

[Azalia Mirhoseini et al: Device Placement Optimization with Reinforcement Learning. **ICML 2017**]

Neural MT graph



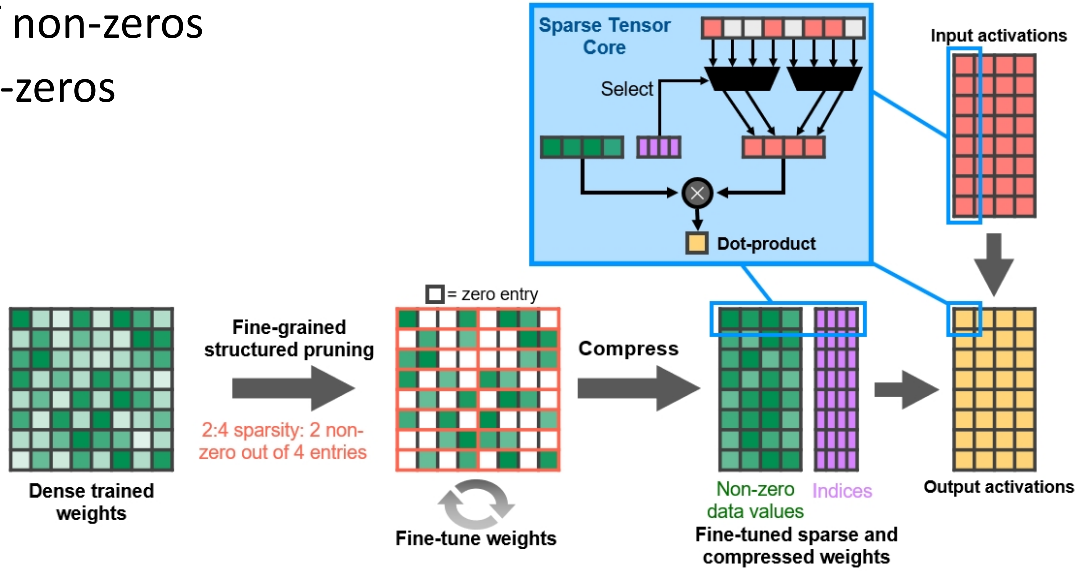
Inception V3



Sparsity in DNN

- **State-of-the-art**
  - **Very limited support of sparse tensors** in TensorFlow, PyTorch, etc
  - GPU operations for linear algebra (**cuSparse**), early support in ASICs
  - Problem: **Irregular structures of sparse matrices/tensors**
- **Common Techniques**
  - #1: **Blocking/clustering** of rows/columns by number of non-zeros
  - #2: **Padding rows/columns** to common number of non-zeros
- **Example A100 Sparsity Exploitation**
  - Constraint: 2 non-zeros in block of 4
  - Structured valued pruning → accuracy impact
  - Regular access pattern

[NVIDIA A100 Tensor Core GPU Architecture, Whitepaper, **Aug 2020**]

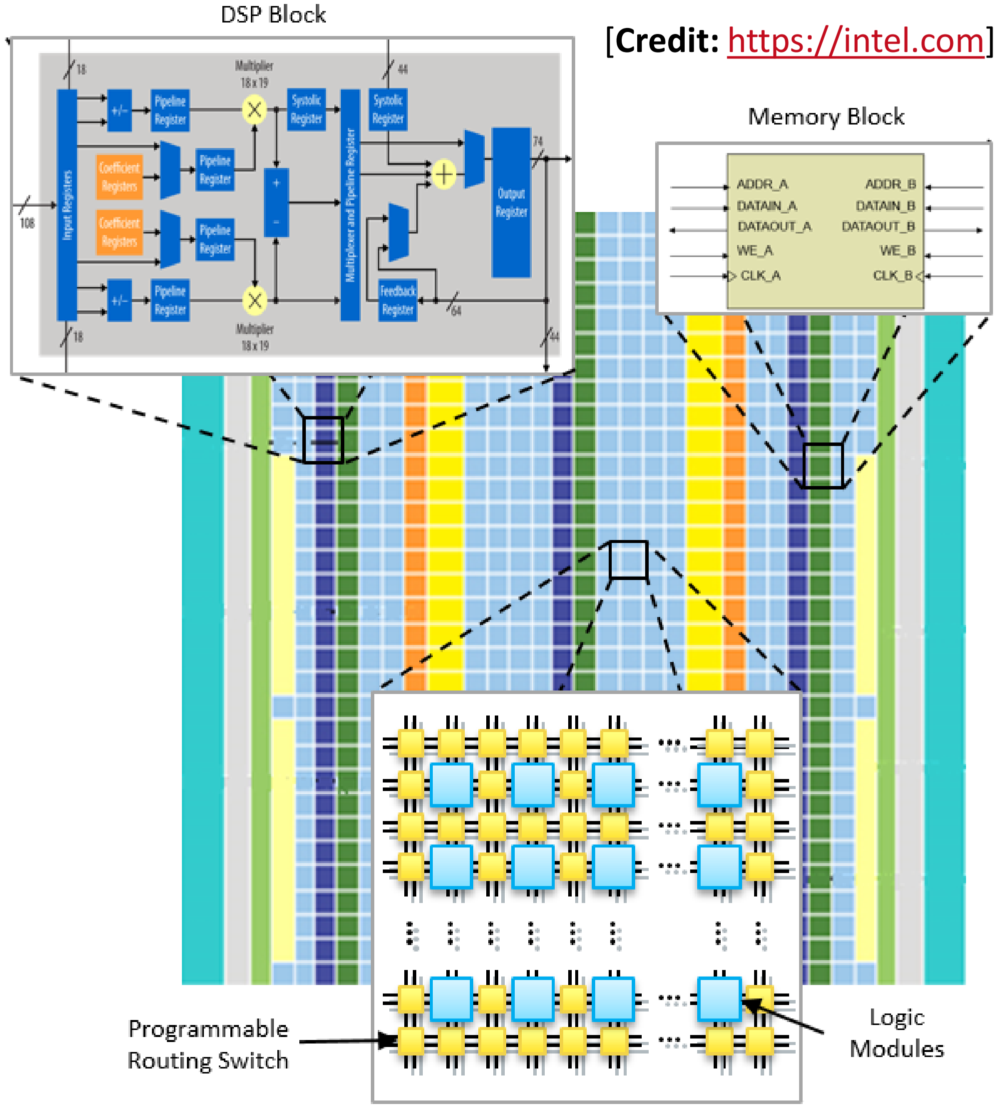


FPGAs in ML Systems

FPGA Overview

- **FPGA Definition**
  - Integrated circuit that allows **configuring custom hardware designs**
  - Reconfiguration in <1s
  - HW description language: e.g., VHDL, Verilog

- **FPGA Components**
  - **#1 lookup table** (LUT) as logic gates
  - **#2 flip-flops** (registers)
  - **#3 interconnect network**
  - Additional memory and DSP blocks



[**Credit:** https://intel.com]

Example FPGA Characteristics

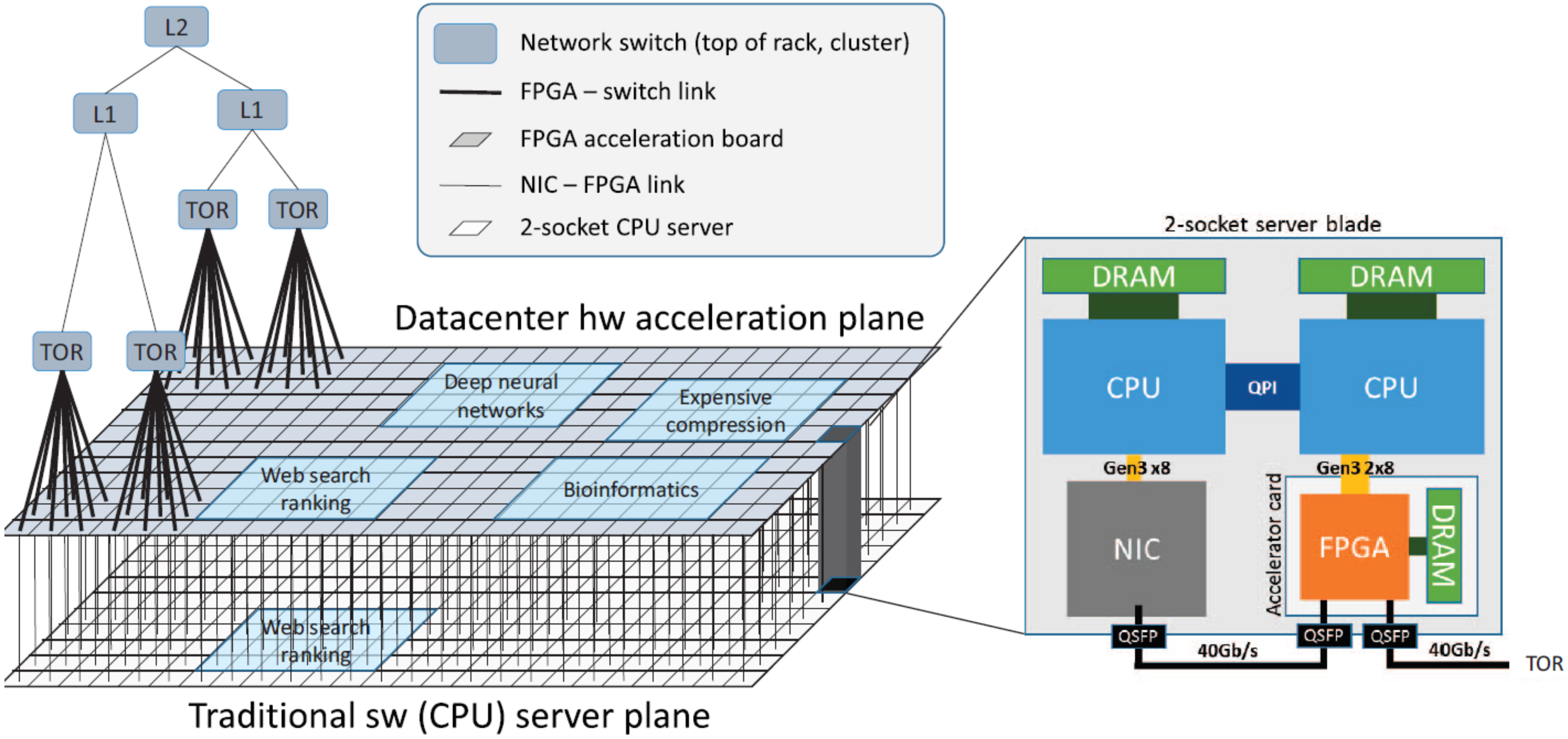
- **Intel (Altera) Stratix 10 SoC FPGA**
  - 64bit quad-core ARM
  - 10 TFLOPs FP32
  - 80GFLOPs/W
  - Other configurations w/ HBM2

- **Xilinx Virtex UltraSCALE+**
  - DSP: 21.2 TMACs
  - 64MB on-chip memory
  - 8GB HBM2 w/ 460GB/s

FPGAs in Microsoft’s Data Centers

[Adrian M. Caulfield et al.: A cloud-scale acceleration architecture. **MICRO 2016**]

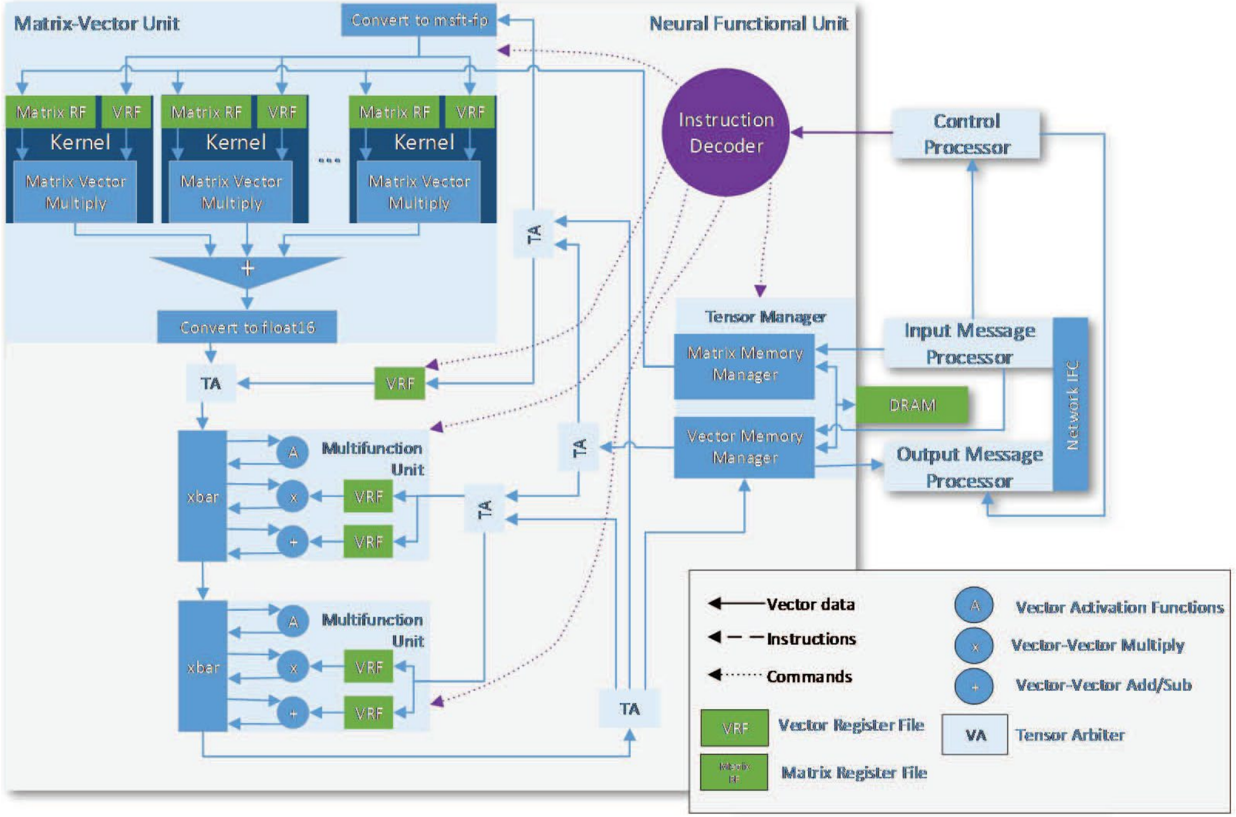
- **Microsoft Catapult**
  - Dual-socket Xeon w/ PCIe-attached FPGA
  - Pre-filtering neural networks, compression, and other workloads



FPGAs in Microsoft's Data Centers, cont.

[Eric S. Chung et al: Serving DNNs in Real Time at Datacenter Scale with Project Brainwave. **IEEE Micro 2018**]

- **Microsoft Brainwave**
  - ML serving w/ low latency (e.g., Bing)
  - Intel Stratix 10 FPGA
  - Distributed **model parallelism**, precision-adaptable
  - Peak 39.5 TFLOPs
- **Brainwave NPU**
  - Neural processing unit
  - Dense matrix-vector multiplication



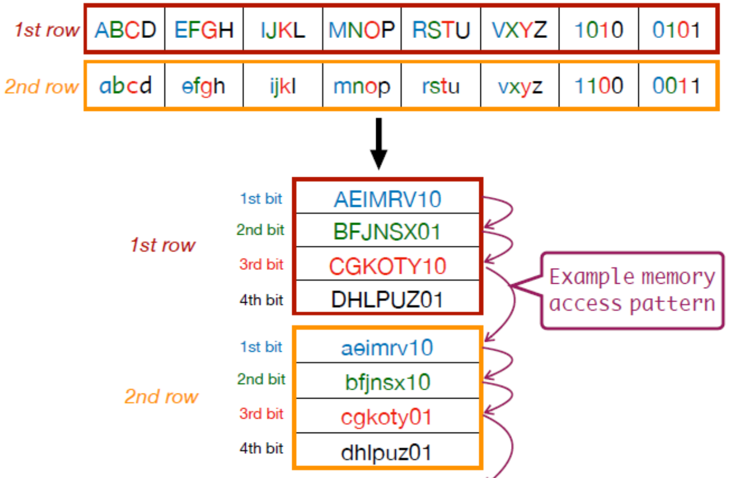
FPGAs in other ML Systems

- **In-DB Acceleration of Advanced Analytics (DAnA)**
  - Compilation of python DSL into micro instructions for multi-threaded FPGA-execution engine
  - Striders to directly **interact with the buffer pool**
- **MLWeaving**
  - Adapted **BitWeaving** to numeric matrices
  - Data layout basis for **Any-Precision Learning**
  - Related FPGA implementation of SGD, matrix-vector multiplication for GLM
  - **Manual Selection** + Heuristics
- **Efficient FPGA implementations of specific operations and algorithms**
- **Specialized neural network topologies**

[Divya Mahajan et al: In-RDBMS Hardware Acceleration of Advanced Analytics. **PVLDB 2018**]

[Zeke Wang et al: Accelerating Generalized Linear Models with MLWeaving. **PVLDB 2019**]

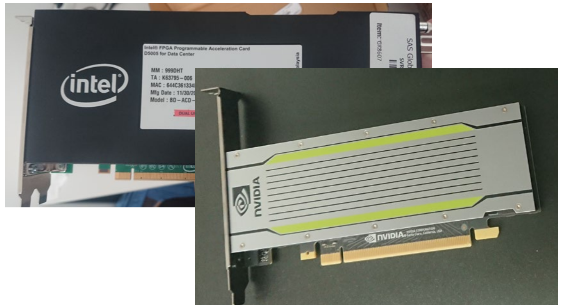
(a) Full-precision fixed-point table TS = T4



(b) BWeaving memory layout

Example DM Cluster Node

- **Setup:** 2x Intel Xeon Gold 6238 (112 vcores, 7.7 TFLOP/s), 768 GB DDR4 RAM, 12x 2TB SSDs, NVIDIA **T4 GPU** (8.1 TFLOP/s, 16 GB), and Intel FPGA PAC D5005 (w/ Stratix **10SX FPGA**, 32 GB)



ASICs and other HW Accelerators

Overview ASICs

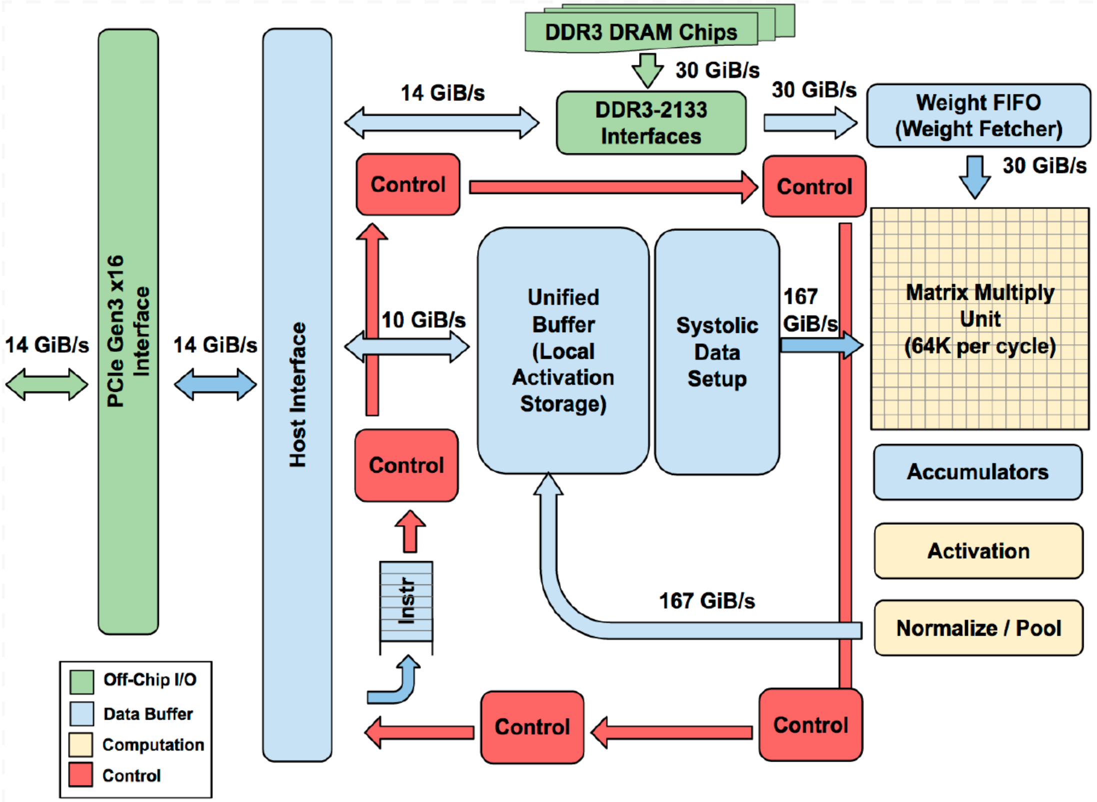
- **Motivation**
  - Additional improvements of performance, power/energy
  - → **Additional specialization via custom hardware**
- **#1 General ASIC DL Accelerators**
  - HW support for matrix multiply, convolution and activation functions
  - Examples: **Google TPU**, **NVIDIA DLA** (in NVIDIA Xavier SoC), **Intel Nervana NNP**
- **#2 Specialized ASIC Accelerators**
  - Custom instructions for specific domains such as computer vision
  - Example: (Cadence) **Tensilica Vision processor** (image processing)
- **#3 Other Accelerators/Technologies** (some skepticism)
  - a) **Neuromorphic computing / spiking neural networks** (e.g., SyNAPSE → IBM TrueNorth, HP memristor for computation storage)
  - b) **Analog computing** (especially for ultra-low precision/quantization)

Tensor Processing Unit (TPU v1)

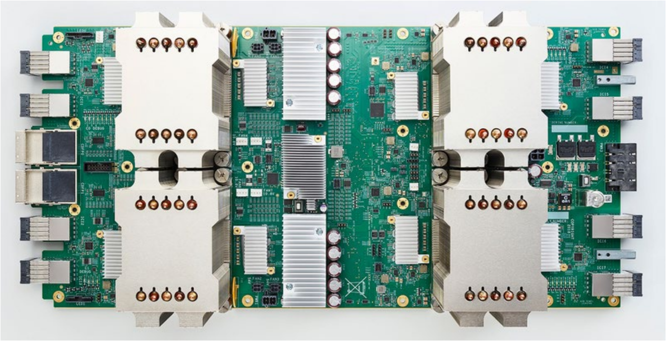
[Norman P. Jouppi et al: In-Datacenter Performance Analysis of a Tensor Processing Unit. **ISCA 2017**]

- **Motivation**
  - Cost-effective ML scoring (no training)
  - Latency- and throughput-oriented
  - **Improve cost-performance over GPUs by 10x**

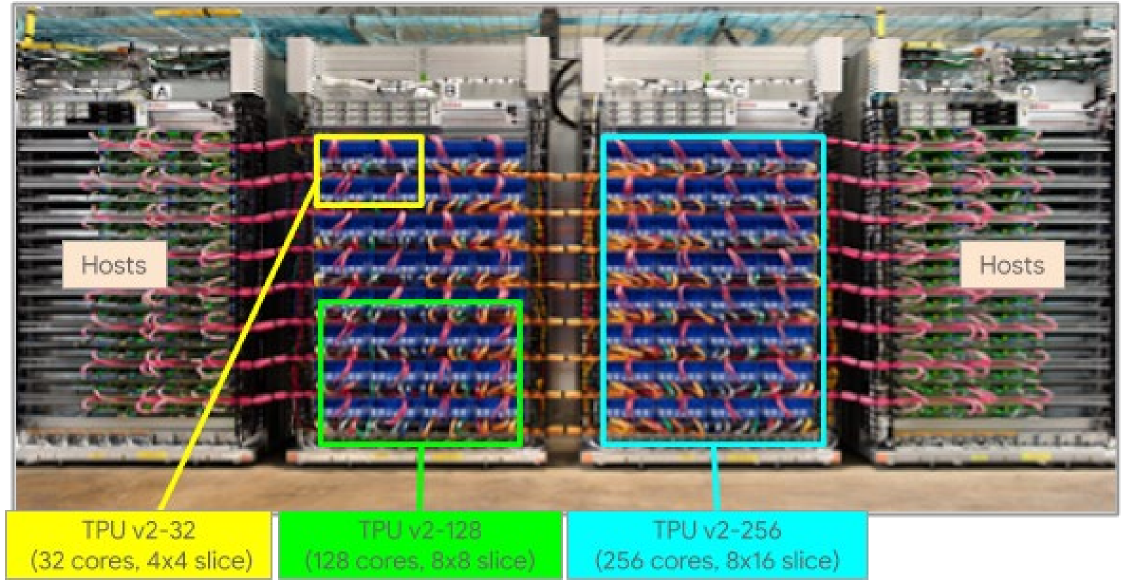
- **Architecture**
  - 256x256 8bit matrix multiply unit (systolic array → **pipelining**)
  - 64K MAC per cycle (92 TOPs at 8 bit)
  - 50% if one input 16bit
  - 25% if all inputs 16 bit



Tensor Processing Unit (TPU v2)

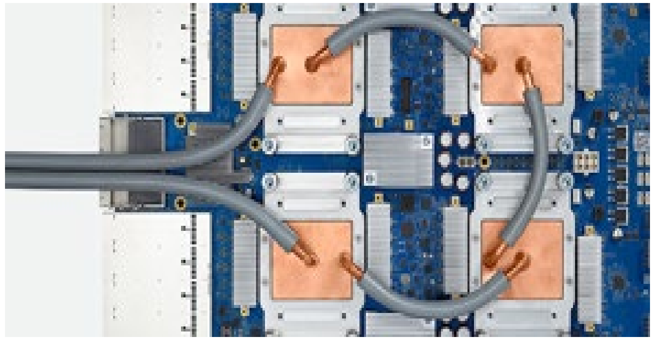


- **Motivation**
  - Cost effective ML training (**not scoring**) because edge device w/ custom inference but training in data centers
  - Unveiled at **Google I/O 2017**
  - Board w/ 4 TPU chips
  - Pod w/ 64 boards and custom high-speed network
  - Shelf w/ 2 boards or 1 processor



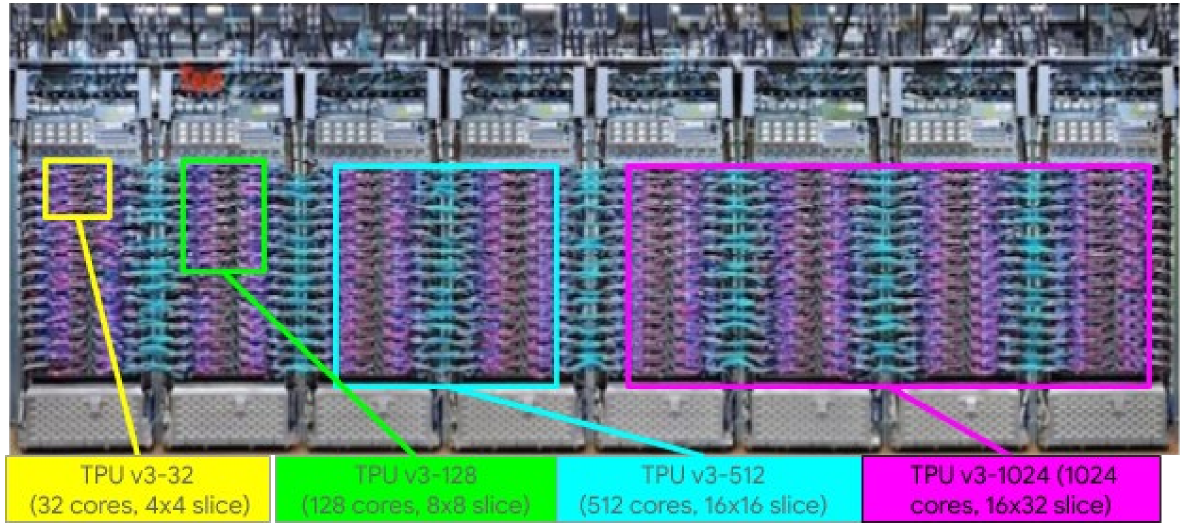
- **Cloud Offering (beta)**
  - Min 32 cores
  - Max 512 cores

Tensor Processing Unit (TPU v3)



- **Motivation**
  - Competitive cost-performance compared to state-of-the-art GPUs
  - Unveiled at **Google I/O 2018**
  - Added **liquid cooling**
  - Twice as many racks per pod, twice as many TPUs per rack

  ➔ TPUv3 promoted as **8x higher performance** than TPUv2



- **Cloud Offering (beta)**
  - Min 32 cores
  - Max 2048 cores (~100PFLOPs)

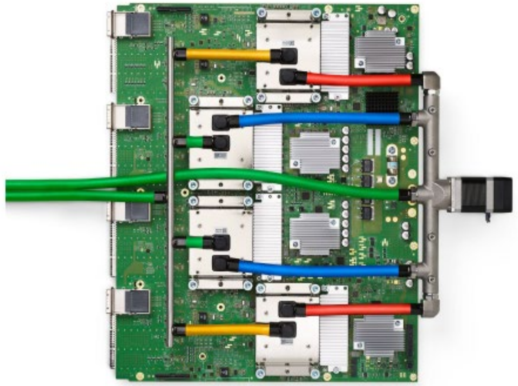
[TOP 500 Supercomputers:
Summit @ Oak Ridge NL ('18):
200.7 PFLOP/s (2.4M cores)]

Tensor Processing Unit (TPU v4)

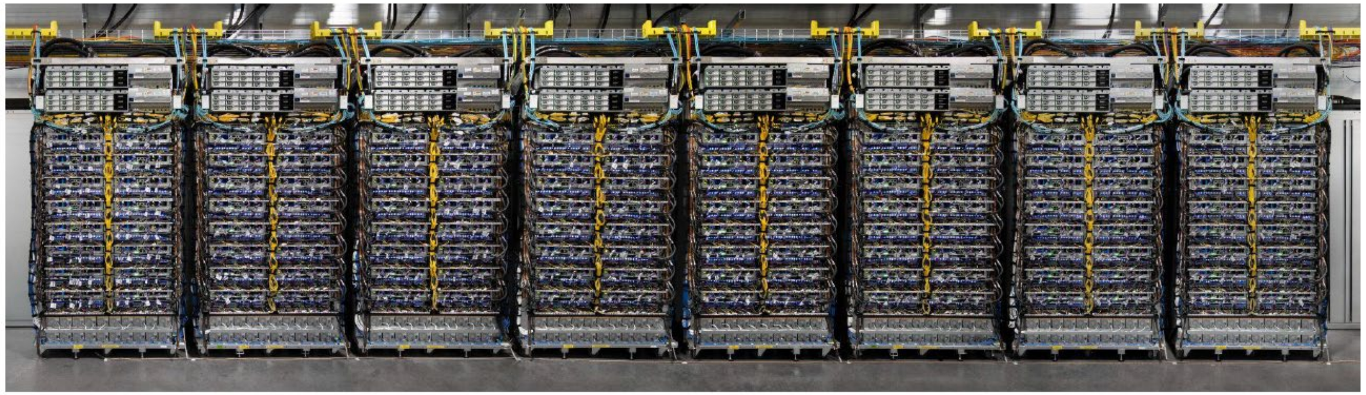
[Norman P. Jouppi et al: TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings. **ISCA 2023**]

[https://cloud.google.com/blog/products/compute/google-unveils-worlds-largest-publicly-available-ml-cluster]

- **Motivation**
  - More chips → twisted 3D torus topology (**reconfigurable optical interconnect**, for fault tolerance)
  - Operational **since 2020**, unveiled at **Google I/O 2021**, paper 2023
  - **SparseCore** (e.g., for sparse gather/scatter)
  - **275 TFLOPs** BF16 or INT8

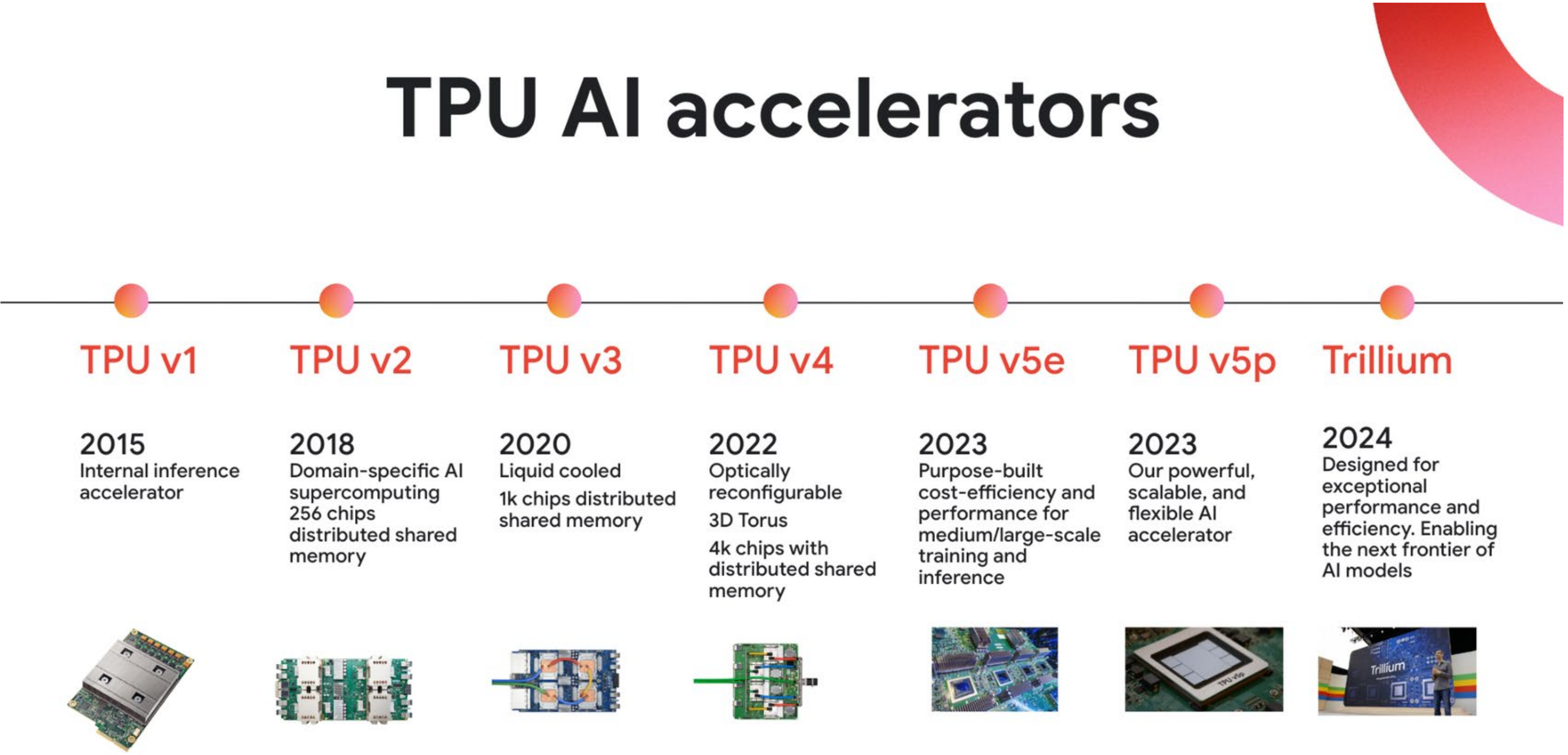


- **Cloud Offering**
  - 4096 chips in 64 racks
  - **1.1 EFLOPs** BF16 or INT8
  - Min 64 chips, max 4096



(8 of 64 racks of a TPUv4 pod)

Tensor Processing Unit over Time



TPU AI accelerators

TPU v1
2015
Internal inference accelerator

TPU v2
2018
Domain-specific AI supercomputing 256 chips distributed shared memory

TPU v3
2020
Liquid cooled
1k chips distributed shared memory

TPU v4
2022
Optically reconfigurable 3D Torus
4k chips with distributed shared memory

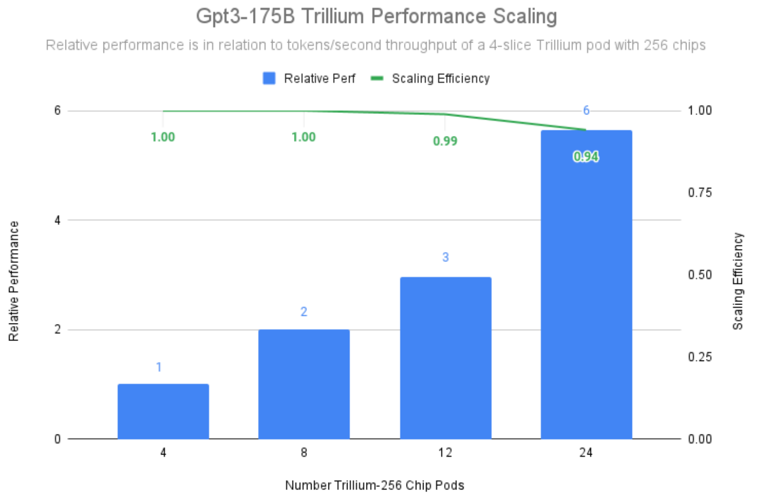
TPU v5e
2023
Purpose-built cost-efficiency and performance for medium/large-scale training and inference

TPU v5p
2023
Our powerful, scalable, and flexible AI accelerator

Trillium
2024
Designed for exceptional performance and efficiency. Enabling the next frontier of AI models

[https://cloud.google.com/transform/ai-specialized-chips-tpu-history-gen-ai]

[https://cloud.google.com/blog/products/ai-machine-learning/introducing-cloud-tpu-v5p-and-ai-hypercomputer]



[https://cloud.google.com/blog/products/compute/trillium-tpu-is-ga]

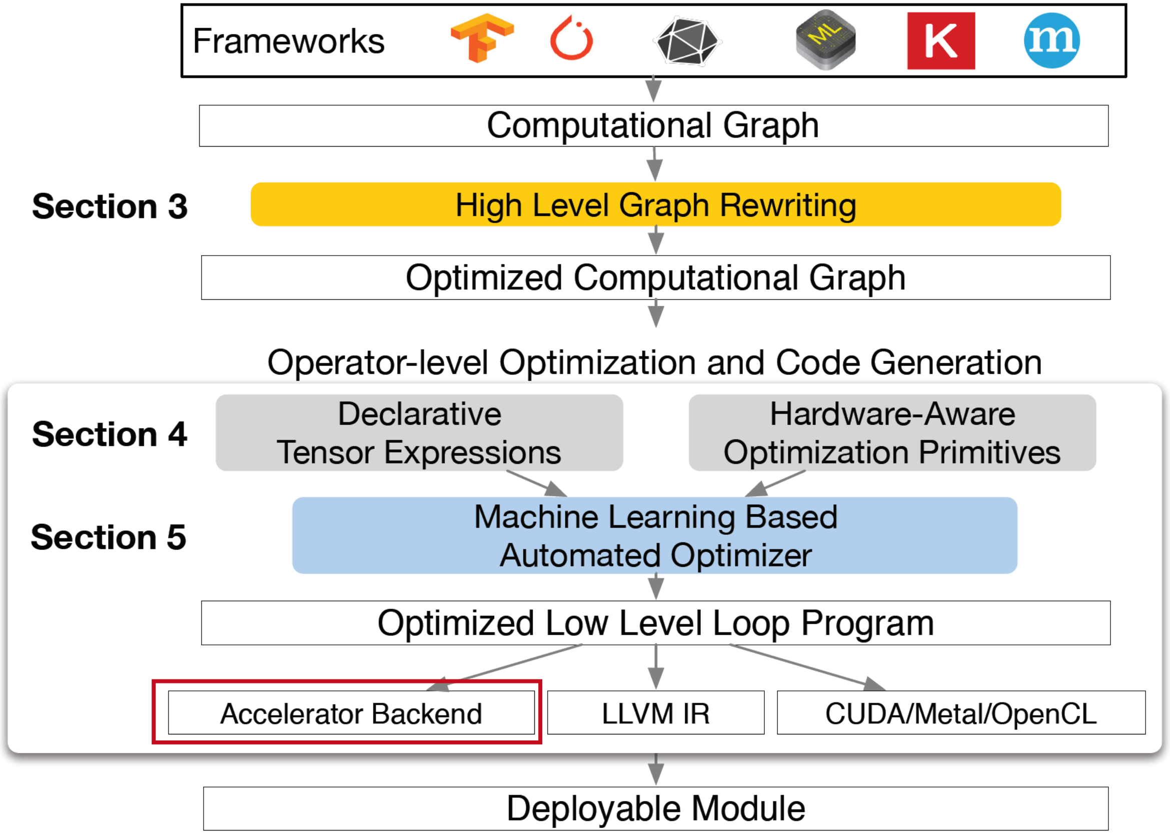
2025 Ironwood: [https://blog.google/products/google-cloud/ironwood-tpu-age-of-inference/]

Recap: Operator Fusion and Code Generation

[Tianqi Chen et al: TVM:
An Automated End-to-End Optimizing
Compiler for Deep Learning. **OSDI 2018**]

- **TVM: Code Generation for HW Accelerators**
  - Graph- /operator-level optimizations for **embedded and HW accelerators**
  - **Lack of low-level instruction set!**
  - Schedule Primitives
    - Loop Transform
    - Thread Binding
    - Compute Locality
    - Tensorization
    - Latency Hiding

➔ **Apache** tvm



Frameworks

Computational Graph

Section 3 High Level Graph Rewriting

Optimized Computational Graph

Operator-level Optimization and Code Generation

Section 4 Declarative Tensor Expressions | Hardware-Aware Optimization Primitives

Section 5 Machine Learning Based Automated Optimizer

Optimized Low Level Loop Program

Accelerator Backend | LLVM IR | CUDA/Metal/OpenCL

Deployable Module

SambaNova

[Kunle Olukotun: Let the Data Flow!, **CIDR 2021**, https://www.youtube.com/watch?v=iHhHHBuk3W4, **SDSC 2020**, https://www.youtube.com/watch?v=E7se0KEa4BY]

Stanford | ENGINEERING
Electrical Engineering
Computer Science

SambaNova SYSTEMS

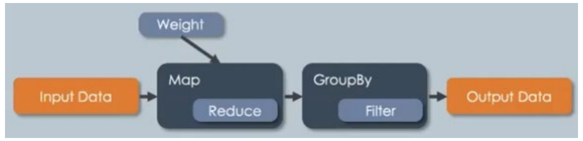
Let the Data Flow!

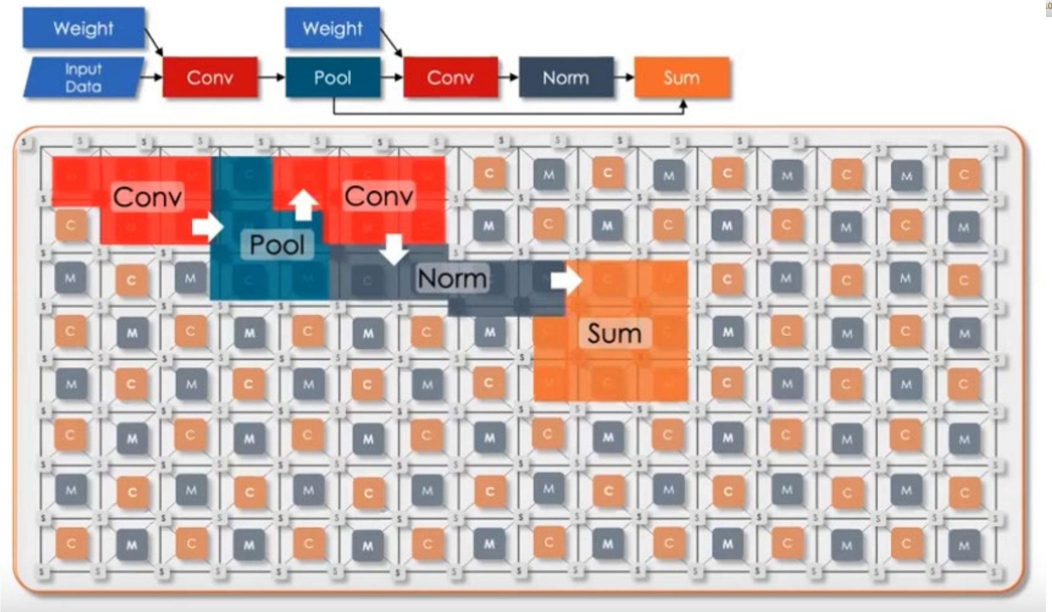
Kunle Olukotun
Stanford University
SambaNova Systems

CIDR Keynote, Jan 14, 2021


DataScale SN10-8R Quarter Rack (10-8, 12TB)
DataScale SN10-8R Half Rack (Includes 2 x SN10-8, 24TB)
DataScale SN10-8R Full Rack (Includes 4 x SN10-8, 48TB)

- **Overview**
  - Reconfigurable data flow architecture
  - Based on **hierarchical parallel patterns** (map, zip, reduce, flatMap, groupBy)
  - Reconfigurable Dataflow Unit (**RDU**), but more coarse-grained than FPGAs
  - 100s of TFLOPs, 100s MB on chip
- **Mapping of Dataflow Computation**
  - DNN / ML
  - Graph processing
  - SQL query processing





reconfigure in ~1-10ms

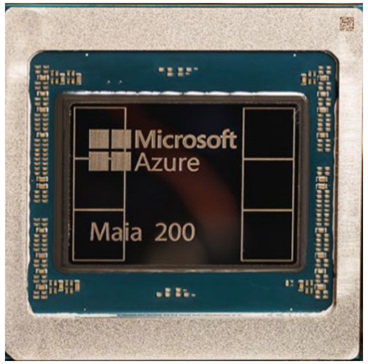
Other Specialized Chips (3nm)

[https://awsdocs-neuron.readthedocs-hosted.com/en/latest/about-neuron/arch/neuron-hardware/neuron-core-v4.html#neuroncores-v4-arch]

- **AWS Trainium Accelerator**
  - Current Generation: Trainium3
  - 8x NeuronCore-v4 (based on systolic arrays):
    - **2,517 MXFP8/MXFP4 TFLOPS**
    - **671 BF16/FP16/TF32 TFLOPS**
    - **2,517 FP16/BF16/TF32 sparse TFLOP**
  - Ultra-Server similar to TPU pods

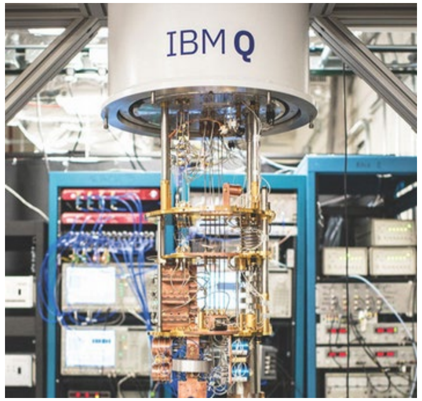
- **Microsoft Maia 200**
  - Dedicated AI inference chip
  - **10 PFLOPS in 4-bit precision (FP4)**
    5 PFLOPS of 8-bit (FP8) performance

[https://blogs.microsoft.com/blog/2026/01/26/maia-200-the-ai-accelerator-built-for-inference/]



Excursus: Quantum Machine Learning

- **Background** (Schrödinger's cat)
  - Concepts: superposition, entanglement, de-coherence / uncertainty
- **IBM Q**
  - Hardware and software stack for quantum computing
  - **Qiskit:** OSS Python framework [https://qiskit.org/]
  - Experiment w/ quantum computers up to 20 qubit
  - **Gates:** Hadamard, NOT, Phases, Pauli, barriers transposed conjugate, if, measurement



- **Early ML (Systems) Work**
  - **Training quantum neural networks** (relied on quantum search in O(√N)
  - **SVM classification** w/ large feature space
  - **TensorFlow Quantum** (TFQ), on OSS **Cirq** for hybrid models [https://www.tensorflow.org/quantum]

[Bob Ricks, Dan Ventura: Training a Quantum Neural Network. **NeurIPS 2003**]

[Vojtěch Havlíček et al: Supervised learning with quantum-enhanced feature spaces. **Nature 2019**]

ML Hardware Fallacies and Pitfalls

- **Recommended Reading**
  - [Jeff Dean, David A. Patterson, Cliff Young: A New Golden Age in Computer Architecture: Empowering the Machine-Learning Revolution. **IEEE Micro 2018**]

- **#1 Fallacy: Throughput over Latency**
  - Given the large size of the ML problems, the HW focus should be op/s (throughput) rather than time to solution (latency)
- **#2 Fallacy: Runtime over Accuracy**
  - Given large speedup, ML researchers would be willing to sacrifice accuracy
- **#3 Pitfall: Designing HW using last year's models**
  - MNIST, CIFAR-10 datasets too easy, AlexNet no longer representative
  - See 02 System Architecture & Landscape – ML System Benchmarks
- **#4 Pitfall: Designing ML HW assuming ML system is untouchable**
  - Towards hardware-software co-design (algorithm, system internals)

Trend: ML-based Chip Placement

- **Motivation**
  - **ASICs:** custom chips for ML
  - ML for improved chip placement (part of chip design process
- **Deep RL for Chip Design**
  - Goal: optimize power, performance, and area **s.t.** constraints on routing congestion and density
  - Approximate reward functions for effective evaluation ~100K (wire length, grid rows/columns, macro order, cell placement, routing congestion)
- **Example TPUv4 Block**
  - White macros (e.g., mem)
  - Green standard cells

[Azalia Mirhoseini, Anna Goldie, et al: Chip Placement with Deep Reinforcement Learning. **CoRR 2020**]

[Azalia Mirhoseini, Anna Goldie, et al: A Graph Placement Methodology for Fast Chip Design. **Nature 2021**]

https://www.youtube.com/watch?v=gSBYf25bWyo

$$R_{p,g} = -\text{Wirelength}(p,g) - \lambda\text{Congestion}(p,g) - \gamma\text{Density}(p,g).$$

Summary & QA

- **Different Levels of Hardware Specialization**
  - General-purpose CPUs and GPUs
  - FPGAs, DNN ASICs, and other technologies

Increasing importance of specialization:
End of Moore's Law
End of Dennard Scaling

- **Next Lectures** (Part A)
  - **09 Caching, Partitioning, Indexing and Compression** [Jun 18]

  - **10 Data Acquisition, Cleaning, and Preparation** [Jun 25]
  - **11 Model Selection and Management** [Jul 02]
  - **12 Model Debugging, Fairness, Explainability** [Jul 09]
  - **13 Model Serving Systems and Techniques** [Jul 16]

  Q&A and Exam Preparation [Jul 16]

(**Part A:**
Overview and ML
System Internals)

(**Part B:**
ML Lifecycle
Systems)