Amazon EC2 Computing and AI Infra

SoonBeom Kwon
AWS Compute Secialist Senior SA

>10 years of silicon innovation

2011 | 2013 | 2015 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 | 2024 | 2025


NITRO V1


NITRO V2


NITRO V3


NITRO V4

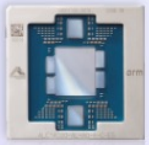

NITRO V5


NITRO V6


GRAVITON


GRAVITON2


GRAVITON3/3E


GRAVITON4

GRAVITON5

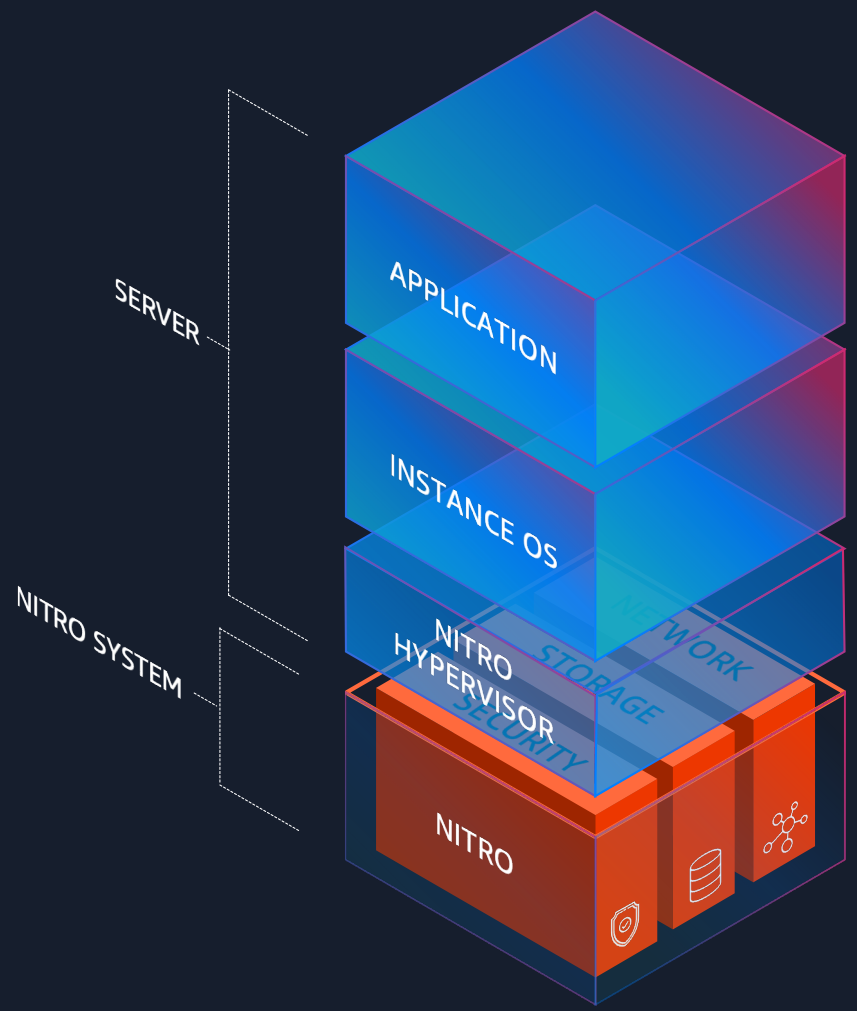

INFERENTIA


TRAINIUM


TRAINIUM2

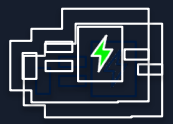
The AWS Nitro System architecture

OFFERING THE BEST SECURITY, PERFORMANCE, AND INNOVATION IN THE CLOUD



AWS Nitro System

Nitro Cards



VPC networking
Block storage
Instance storage
System controller

Nitro Hypervisor

Lightweight hypervisor
Memory and CPU allocation
Bare metal–like performance

Nitro Security Chip

Integrated into motherboard
Traps I/O to nonvolatile storage
Hardware root of trust

Broad and deep accelerated computing portfolio

GPU, AWS ML ACCELERATORS, AND FPGA-BASED EC2 INSTANCES

aws
Trainium accelerator
Inferentia accelerator

NVIDIA
B200, H200, H100, A100, L4, L40S
A10G, T4

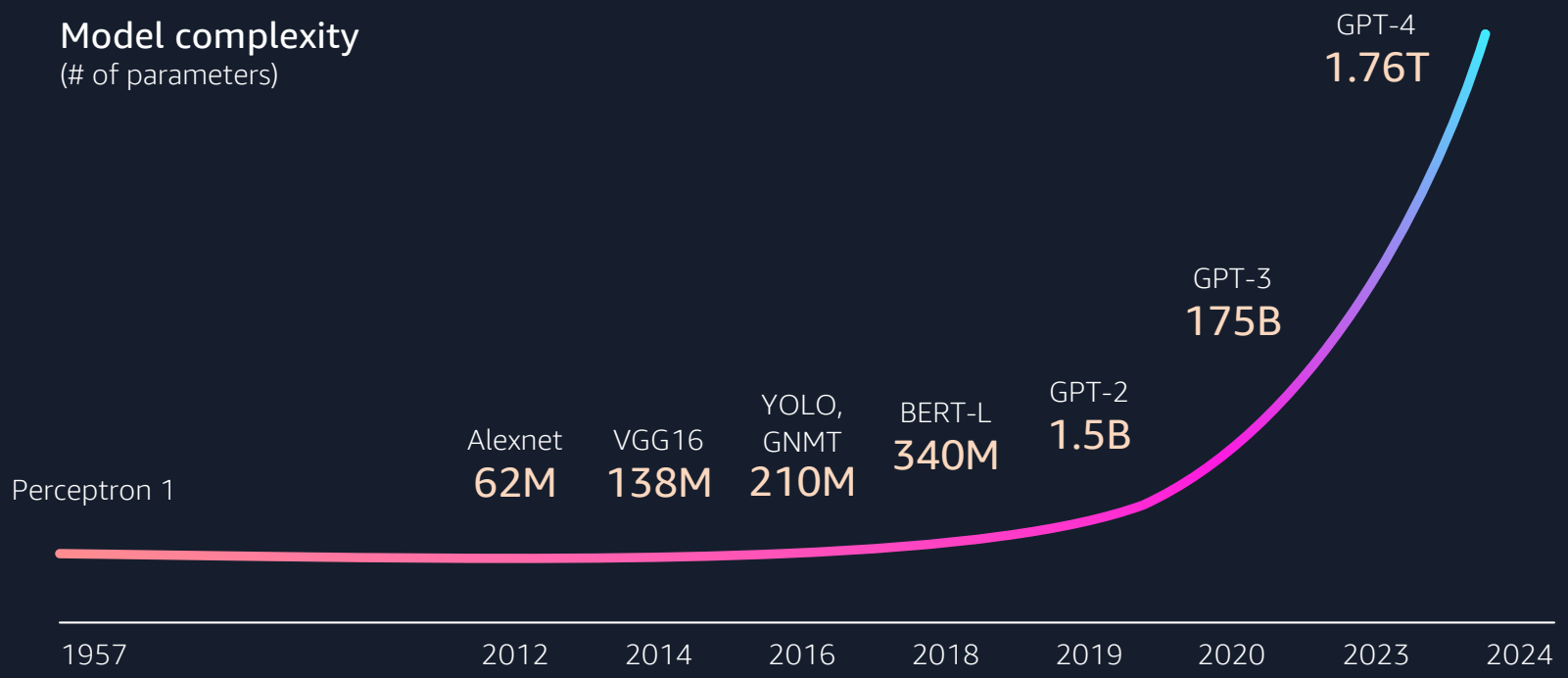
intel
Gaudi accelerator

AMD
Radeon GPU
Xilinx accelerator
Xilinx FPGA

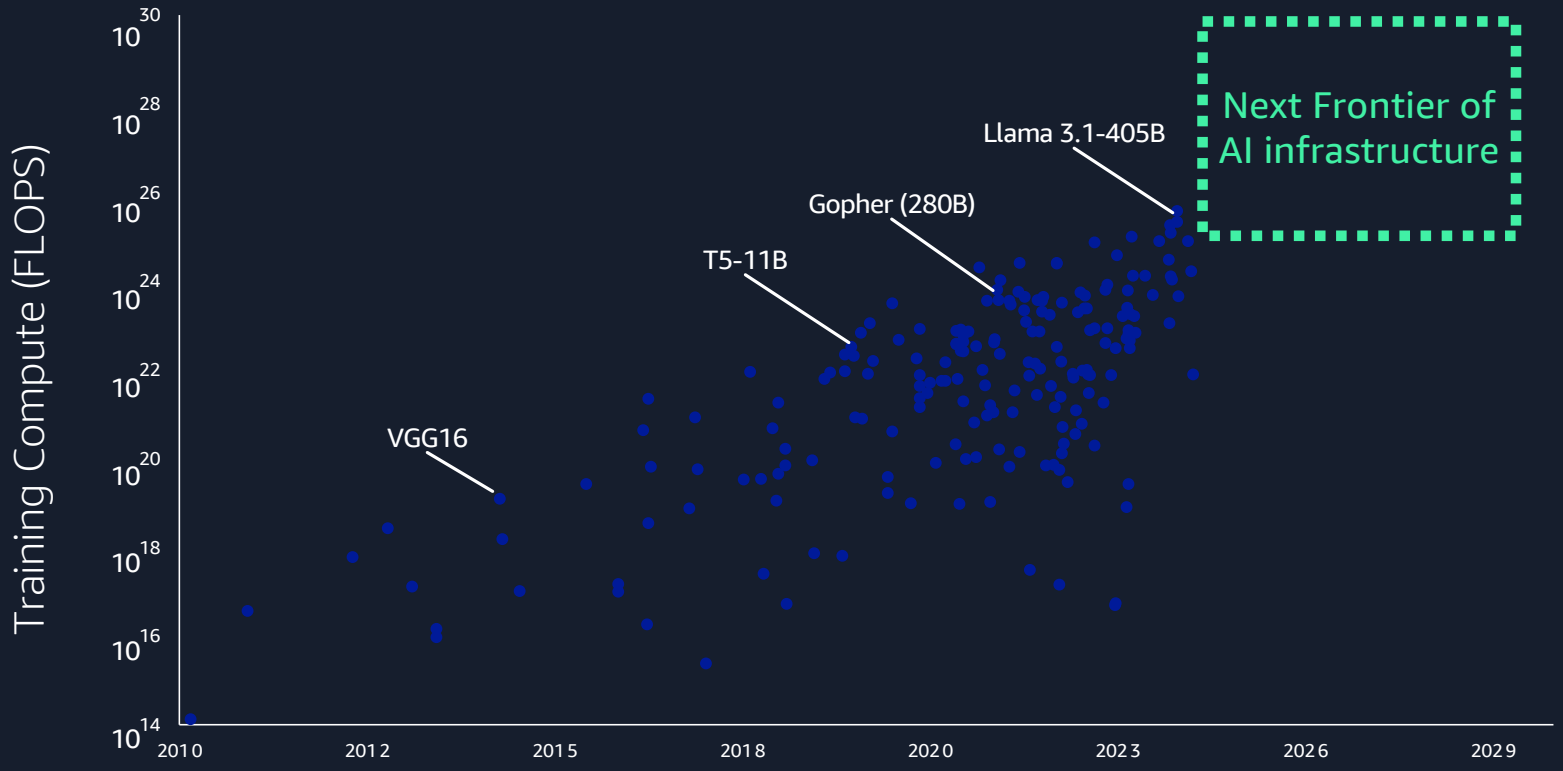
Qualcomm
Qualcomm Cloud AI 100

Machine Learning at scale

Model scaling drives silicon and datacenter scale



Scaling compute requires next-generation AI infrastructure



Source: Epoch AI, 'Data on Notable AI Models'. 26 Nov 2024

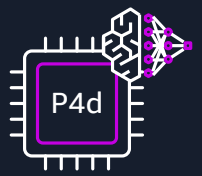
Overview of AI/ML infrastructure

EC2 instances aws NVIDIA AMD intel Qualcomm

ENHANCED SECURITY AND PERFORMANCE WITH AWS NITRO

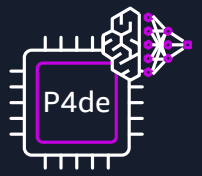
COMPATIBLE WITH:	FULL STACK OPTIMIZATIONS:	AVAILABLE THROUGH:
ML frameworks & open source	**Storage/networking**	**Managed services**
PyTorch	AWS EFA	Amazon SageMaker / HyperPod
TensorFlow	Amazon S3	Amazon EKS
Hugging Face	Amazon FSx for Lustre	AWS ParallelCluster (EC2)
JAX	Amazon EC2 UltraClusters	
	Amazon EC2 UltraServers	

Accelerated computing for ML training



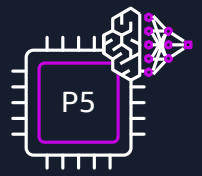
P4d:
NVIDIA A100 * 8

- 2.5 petaflops FP16 compute and 320 GB HBM2
- Up to 400 Gbps networking (EFA) and 600 GB/s device-device interconnect
- GPU Memory bandwidth 1.6TB/s
- 156 teraflops FP32



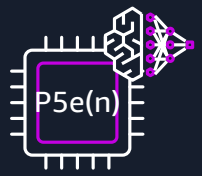
P4de:
NVIDIA A100 * 8

- 2.5 petaflops FP16 compute and 640 GB HBM2e
- Up to 400 Gbps networking (EFA) and 600 GB/s device-device interconnect
- GPU Memory bandwidth 2TB/s
- 156 teraflops FP32



P5:
NVIDIA H100 * 8

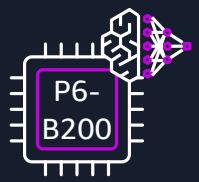
- 8 petaflops FP16 compute and 640 GB HBM3
- 3,200 Gbps EFAv2 networking and 900 GB/s device-device interconnect
- GPU Memory bandwidth 3.35 TB/s
- 536 teraflops FP32



P5e(n):
NVIDIA H200 * 8

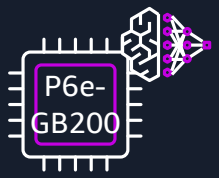
- 8 petaflops FP16 compute and up to 1146 GB HBM3e
- 3,200 Gbps EFAv3 networking and 900 GB/s device-device interconnect
- GPU Memory bandwidth 4.8 TB/s
- 536 teraflops FP32

Accelerated computing for ML training



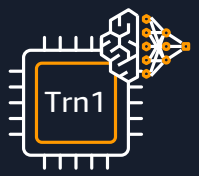
P6-B200:
NVIDIA B200

- Up to 18 petaflops FP16 compute and up to 1440 GB HBM3e
- Up to 3,200 Gbps networking (EFAv4) and 1800 GB/s device-device interconnect



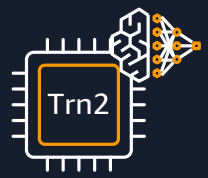
P6e-GB200:
NVIDIA GB200

- Up to 180 petaflops FP16 compute and up to 13320 GB HBM3e
- Up to 28,000 Gbps networking (EFAv4)
- Ultraservers with 36 to 72 GPUs under NVLink



Trn1:
AWS Trainium1

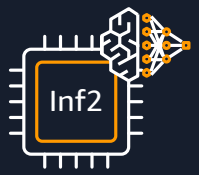
- Up to 3.4 petaflops FP16 compute and 512 GB HBM
- Up to 1,600 Gbps networking (EFA) and 768 Gib/s device-device interconnect



Trn2:
AWS Trainium2

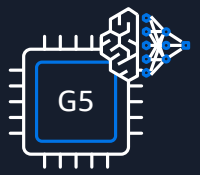
- Up to 10.4 petaflops FP16 compute and 1.5 TB HBM3
- Up to 3,200 Gbps EFAv3 networking and 1 Tb/s chip-chip interconnect

Accelerated computing for ML Inference



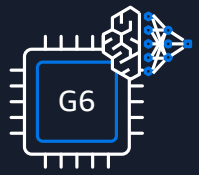
Inf2:
AWS Inferentia2

- Up to 12 Inf2 Chips
- Up to 384 GB vRAM @ 600 Gb/s



G5:
NVIDIA A10G

- Up to 8 NVIDIA A10G GPUs
- Up to 192 GB vRAM @ 600 GB/s



G6:
NVIDIA L4

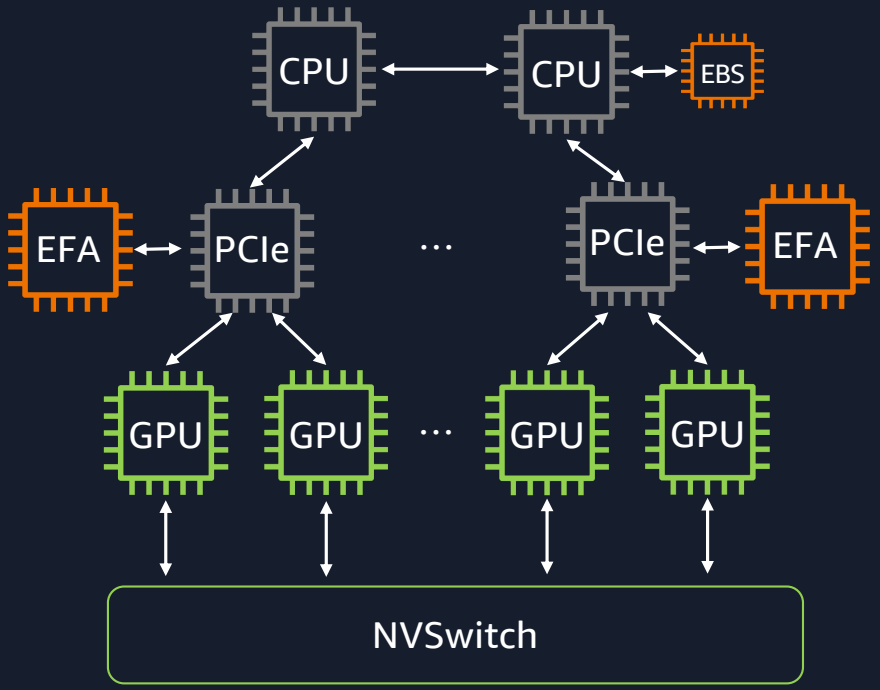
- Up to 8 NVIDIA L4 GPUs
- Up to 192 GB vRAM @ 300 GB/s

G6e:
NVIDIA L40S

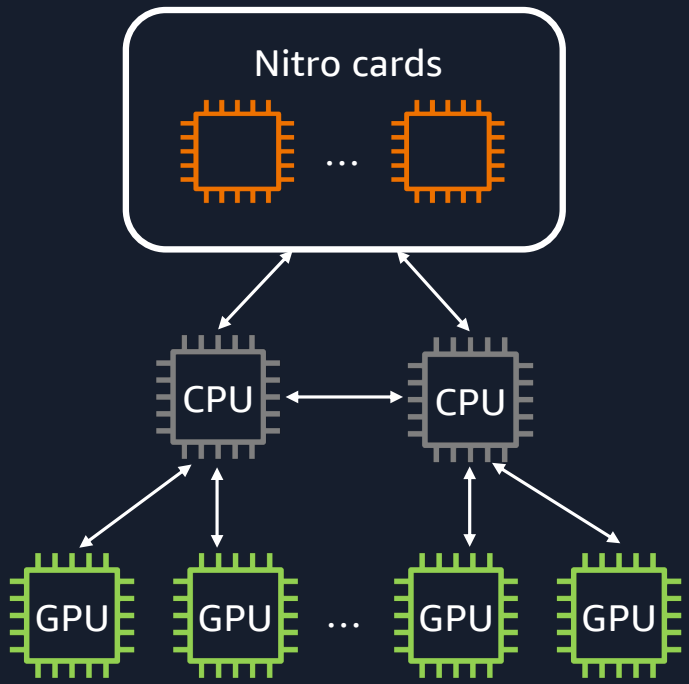
- Up to 8 NVIDIA L40S GPUs
- Up to 384 GB vRAM @ 864 GB/s and PCIe Gen4 x16 64 GB/s interconnect
- Up to 2.82 petaflops FP16 Compute
- Up to 732 teraflops FP32 Up to 5.72 petaflops FP8 Compute

Instance architectures

P instance architecture

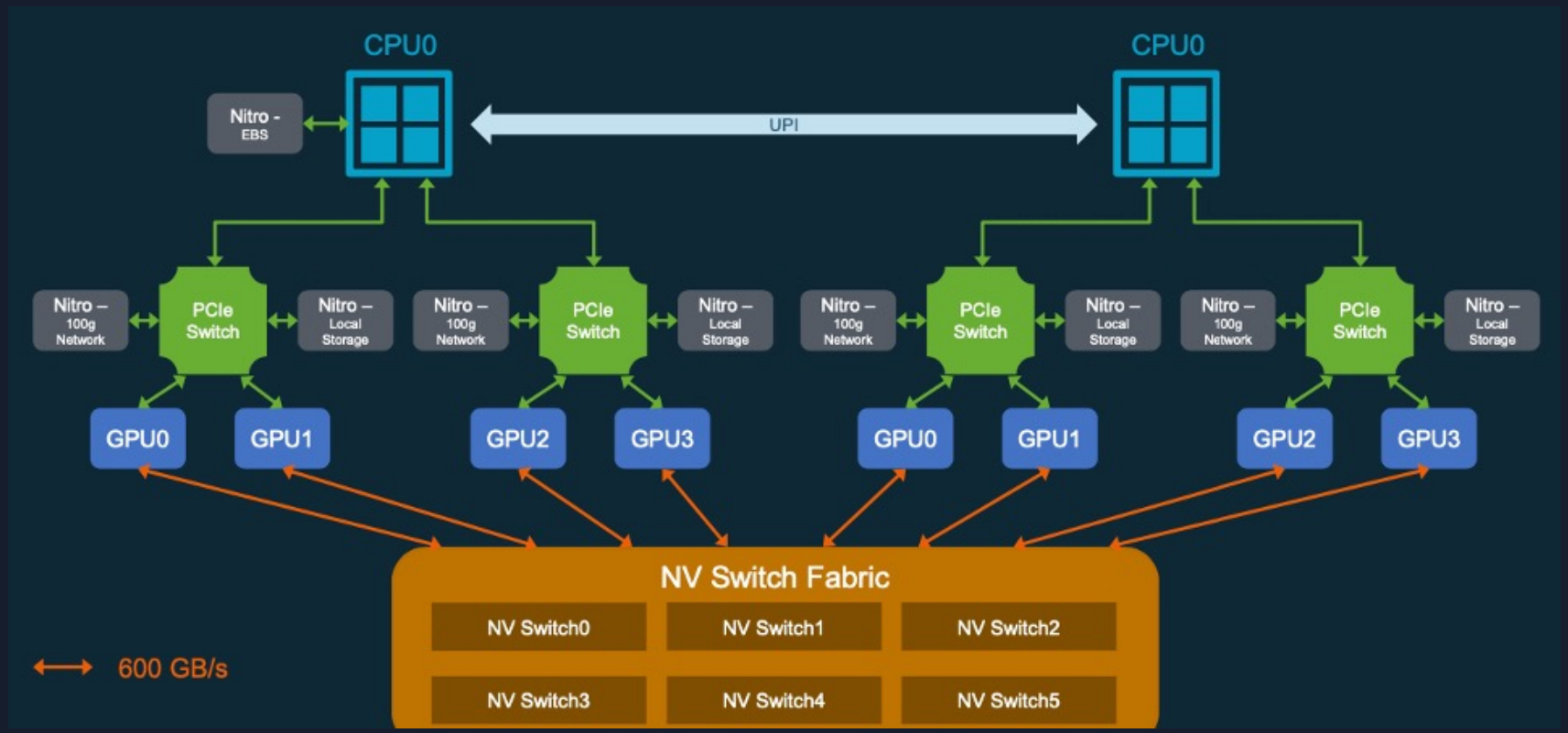


G instance architecture



[AWS re:Invent 2023 - Cutting-edge AI with AWS and NVIDIA (CMP208)]

P4d Instance HardWare Architecture



AWS Blogs - Amazon EC2 P4d instances deep dive

GPU Topology

NVSwitch/Link

P2P/CUMEM/read (polling)
P2P/CUMEM (push)

Bandwidth 600GB/s ~

PCIe

SYS / NODE / PHB /
PXB / PIX

BUS bandwidth loss due to other PCI Devices and latency by CPU intervention

Up to 64GB/s Bandwidth

EFA

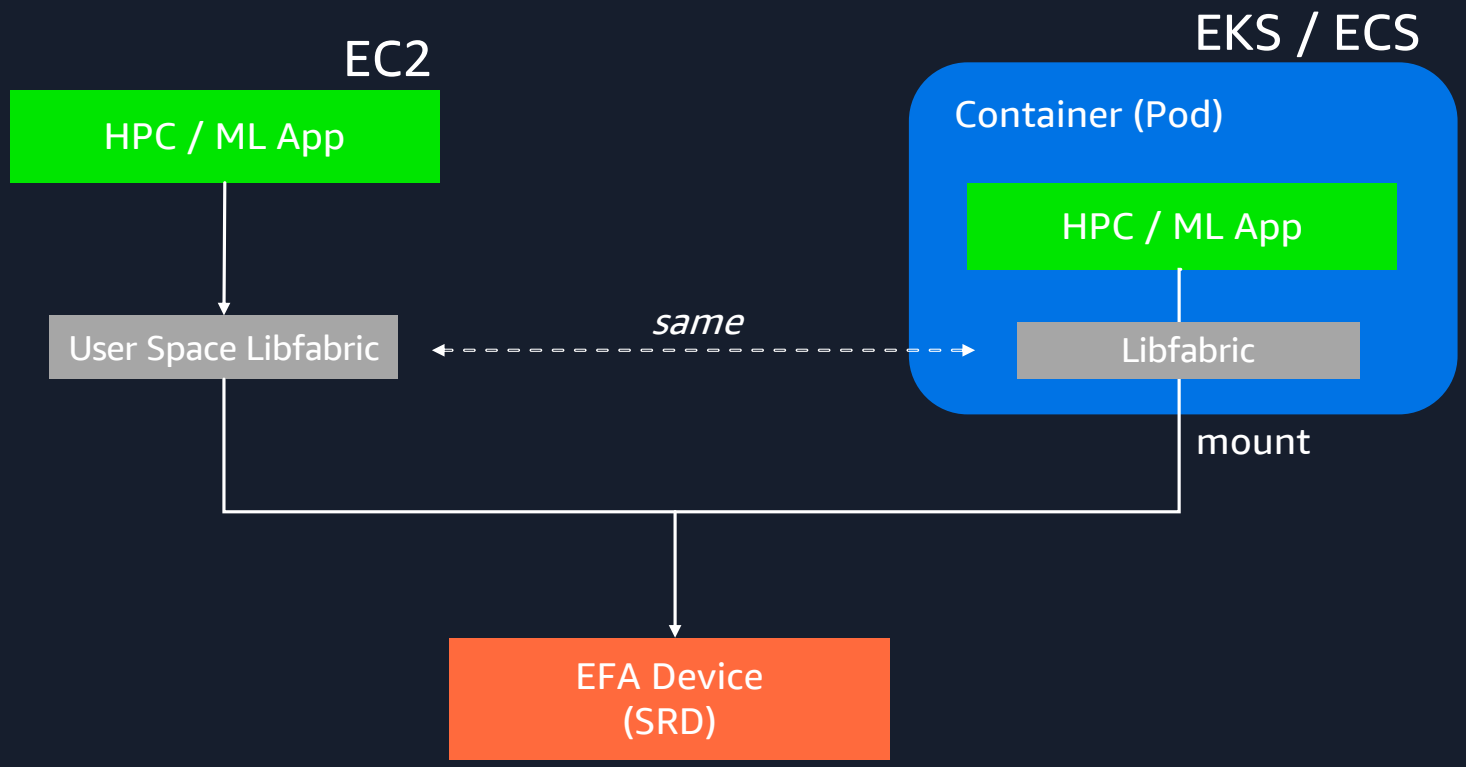
RDMA
GPU Direct RDMA

Bandwidth 100Gbps
Per NIC ~

ENI / ENA

TCP/IP Socket

EC2 / Container EFA Pass Through



GPU Training Time By Topology (world size 4)

Llama-3-8B training with Huggingface + DeepSpeed Stage 3 **within Single Node**

DeepSpeed Config

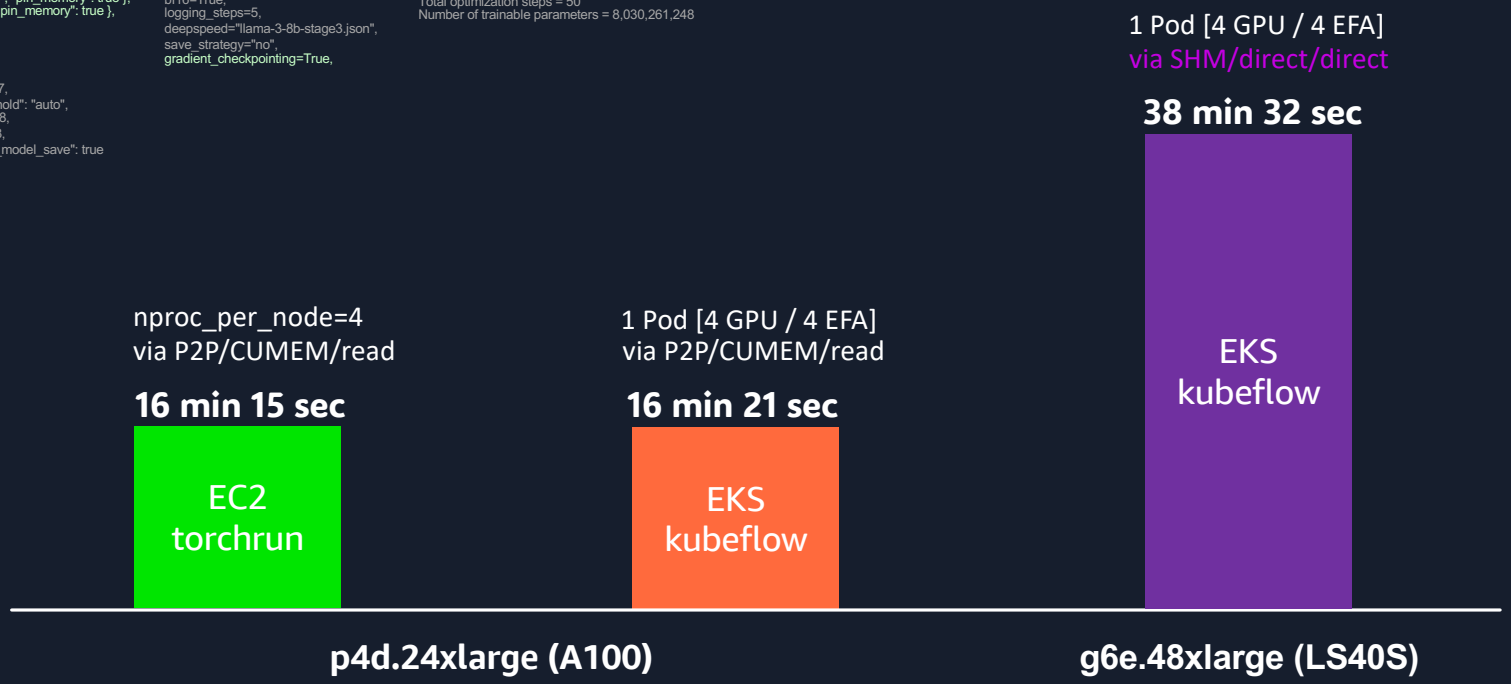
```
{
  "fp16": { "enabled": false },
  "bf16": { "enabled": true },
  "zero_optimization": {
    "stage": 3,
    "offload_optimizer": { "device": "cpu", "pin_memory": true },
    "offload_param": { "device": "cpu", "pin_memory": true },
    "overlap_comm": true,
    "contiguous_gradients": true,
    "sub_group_size": 1e8,
    "reduce_bucket_size": 5e7,
    "stage3_prefetch_bucket_size": 5e7,
    "stage3_param_persistence_threshold": "auto",
    "stage3_max_live_parameters": 1e8,
    "stage3_max_reuse_distance": 1e8,
    "stage3_gather_16bit_weights_on_model_save": true
  },
  "gradient_clipping": "auto",
  "steps_per_print": 10,
  "train_micro_batch_size_per_gpu": 1,
  "gradient_accumulation_steps": 4,
  "wall_clock_breakdown": false
}
```

Trainer Config

```
per_device_train_batch_size=1,
gradient_accumulation_steps=4,
learning_rate=2e-5,
max_steps=50,
num_train_epochs=1,
bf16=True,
logging_steps=5,
deepspeed="llama-3-8b-stage3.json",
save_strategy="no",
gradient_checkpointing=True,
```

******* Running training *******

```
Num examples = 36,718
Num Epochs = 1
Instantaneous batch size per device = 1
Total train batch size (w. parallel, distributed & accumulation) = 16
Gradient Accumulation steps = 4
Total optimization steps = 50
Number of trainable parameters = 8,030,261,248
```



https://github.com/gnosia93/training-on-eks

GPU Training Time By Topology (world size 4)

Llama-3-8B training with Huggingface + DeepSpeed Stage 3 **with Multiple Node**

DeepSpeed Config

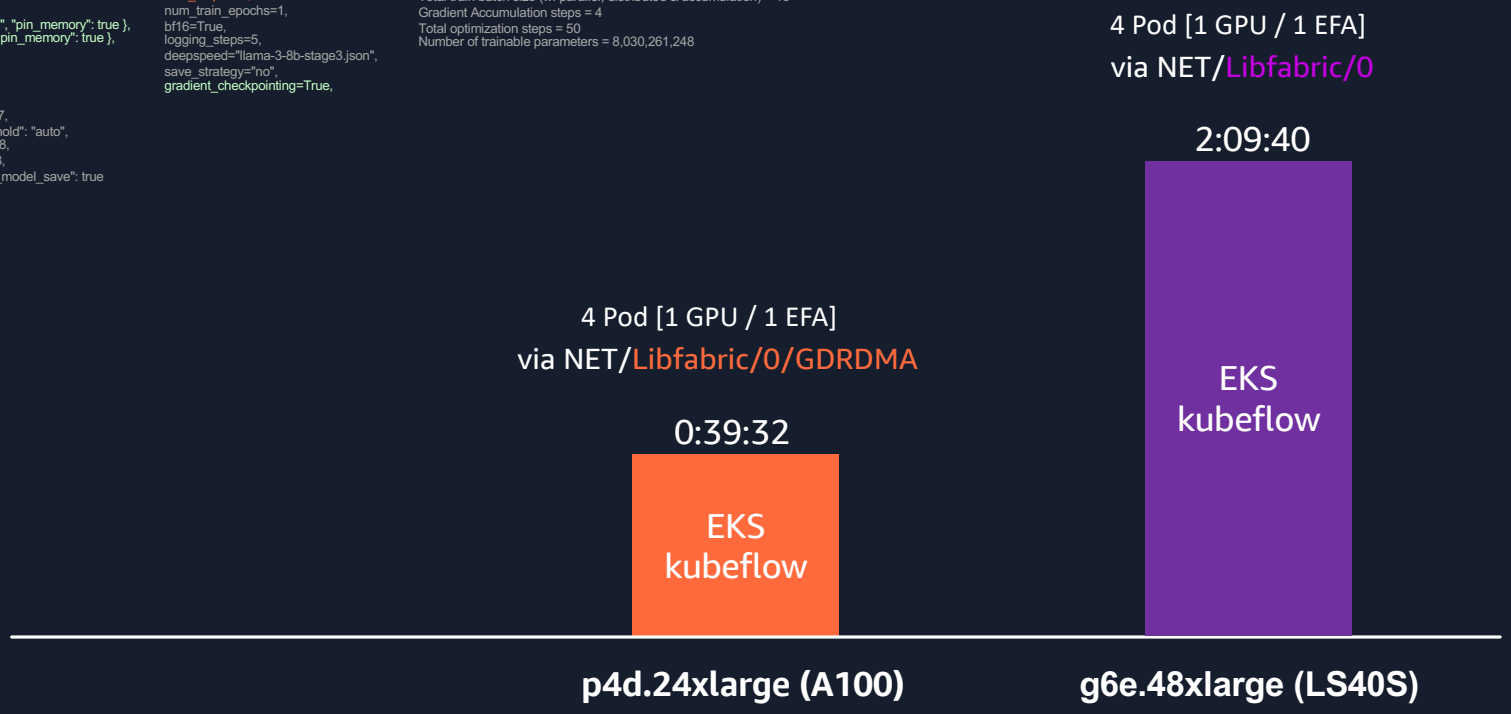
```
{
  "fp16": { "enabled": false },
  "bf16": { "enabled": true },
  "zero_optimization": {
    "stage": 3,
    "offload_optimizer": { "device": "cpu", "pin_memory": true },
    "offload_param": { "device": "cpu", "pin_memory": true },
    "overlap_comm": true,
    "contiguous_gradients": true,
    "sub_group_size": 1e8,
    "reduce_bucket_size": 5e7,
    "stage3_prefetch_bucket_size": 5e7,
    "stage3_param_persistence_threshold": "auto",
    "stage3_max_live_parameters": 1e8,
    "stage3_max_reuse_distance": 1e8,
    "stage3_gather_16bit_weights_on_model_save": true
  },
  "gradient_clipping": "auto",
  "steps_per_print": 10,
  "train_micro_batch_size_per_gpu": 1,
  "gradient_accumulation_steps": 4,
  "wall_clock_breakdown": false
}
```

Trainer Config

```
per_device_train_batch_size=1,
gradient_accumulation_steps=4,
learning_rate=2e-5,
max_steps=50,
num_train_epochs=1,
bf16=True,
logging_steps=5,
deepspeed="llama-3-8b-stage3.json",
save_strategy="no",
gradient_checkpointing=True,
```

******* Running training *******

```
Num examples = 36,718
Num Epochs = 1
Instantaneous batch size per device = 1
Total train batch size (w. parallel, distributed & accumulation) = 16
Gradient Accumulation steps = 4
Total optimization steps = 50
Number of trainable parameters = 8,030,261,248
```

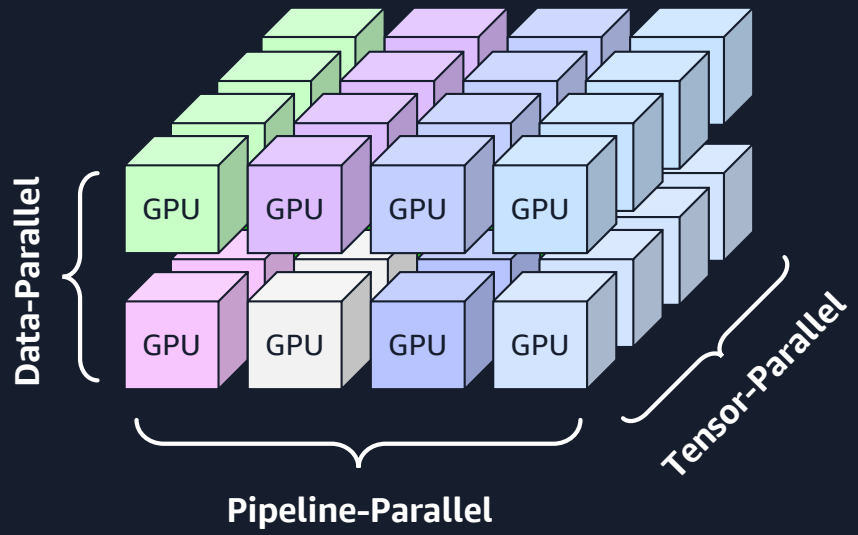


https://github.com/gnosia93/training-on-eks

LLM Workload Chracteristics

Pretraining/Finetuning	Inference
• Network-Intensive • Memory-Bound • I/O Bound • Compute-Bound	• Memory-Bound
Recommendations • P4/P5/P6 • EC2 Ultraclusters	Recommendations • P4/G6 • Spread Placement

3D-Parallelism to scale GPU workloads



GPU Communication Cost by Distributed Training Strategy

Strategy	Primary Communication Collective	Communication Overhead & Cost	Key Characteristics
DDP	**All-Reduce**	**High** (Synchronizes all gradients per step)	Best for models fitting in a single GPU. Scaling is limited by gradient size.
FSDP	**All-Gather / Reduce-Scatter**	**Moderate to High** (Sharded states reduce peak bandwidth pressure)	Memory-efficient by sharding parameters. More scalable than DDP for Large Language Models (LLMs).
TP	**All-Reduce**	**Extremely High** (Communication happens within every layer)	Requires ultra-high bandwidth (e.g., **NVLink**). Highly sensitive to latency.
PP	**P2P (Point-to-Point)**	**Low** (Only at micro-batch boundaries)	High efficiency across nodes but suffers from "pipeline bubbles" (GPU idle time).
MoE	**All-to-All**	**Very High** (Routing tokens to expert nodes)	Generates massive traffic when experts are distributed across different nodes.

- **TP** and **MoE** are highly sensitive to **network latency**, making them ideal for intra-node.
- **FSDP** is often preferred over DDP for large-scale clusters because it overlaps communication with computation more effectively, reducing the "stop-and-wait" bottleneck.
- **7B ~ 13B (Small to Mid) – FSDP, 30B ~ 40B (The Tipping Point): FSDP + TP,** 70B and above (Large): 3D Parallelism is required

The Causes of GPU Starvation

1. Data Pipeline Bottlenecks

This is the most frequent cause, where the CPU or storage cannot supply data as fast as the GPU can process it.

- **Heavy Pre-processing:** Complex tasks like image decoding, heavy data augmentation, or tokenization performed on the CPU can become a bottleneck.
- **I/O Latency:** High-latency storage systems (slow HDDs or remote cloud storage) fail to maintain the required throughput for high-performance GPUs
- **Insufficient Data Workers:** Using too few CPU threads for data loading prevents the data pipeline from staying ahead of the GPU.

2. Communication Overhead & Synchronization

In distributed environments, GPUs must exchange gradients or model parameters, leading to "communication stalls".

- **Network Bandwidth:** If the interconnect is too slow, GPUs spend more time waiting for AllReduce operations to finish than actually computing.
- **The "Straggler" Effect:** In synchronous training, the entire cluster must wait for the slowest GPU to finish its step. If one node has a hardware issue or a background process interfering, every other GPU stays idle.
- **Barrier Synchronization:** Frequent sync points in algorithms like **Pipeline Parallelism** can create "bubbles" (idle periods) where GPUs wait for the next micro-batch to arrive from a previous stage.

3. Hardware and Configuration Limits

- **Small Batch Sizes:** If the per-GPU batch size is too low, the GPU finishes the computation too quickly, making the constant overhead of starting new kernels and loading data more visible.
- **PCIe/NVLink Saturation:** Limited bandwidth between the CPU and GPU (PCIe) or between GPUs (NVLink) can throttle the data flow, keeping the compute cores underutilized.
- **Memory Constraints:** If a model nearly exhausts GPU memory, aggressive memory management or swapping data to host memory can cause significant execution delays.

Elastic Fabric Adapter (EFA)

SRD protocol

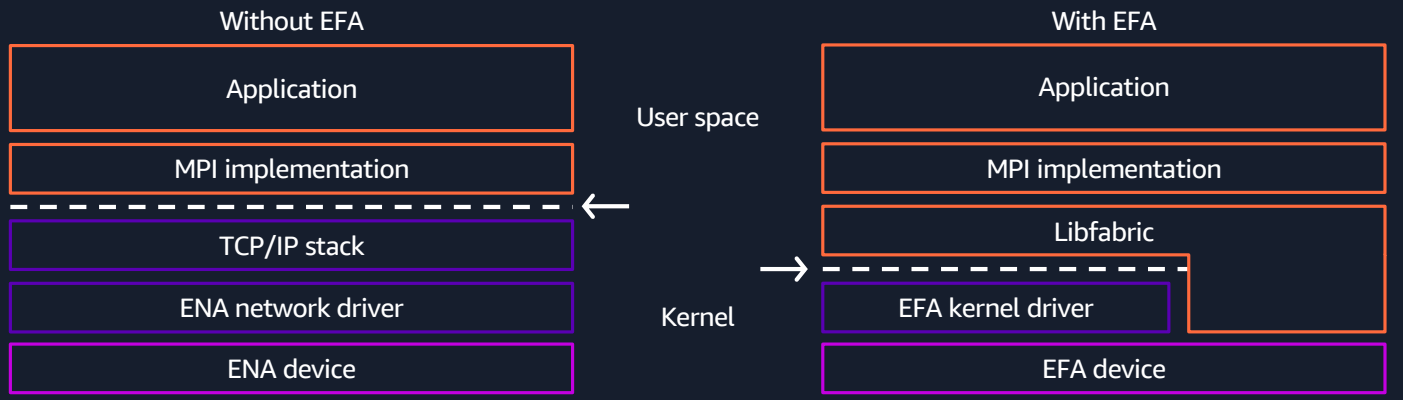
User Datagram Protocol
Multi Pass Routing /
Congestion Control

Provide High Performance Networking for AI and HPC

Nitro Card for VPC
OS bypass
Zero Copy

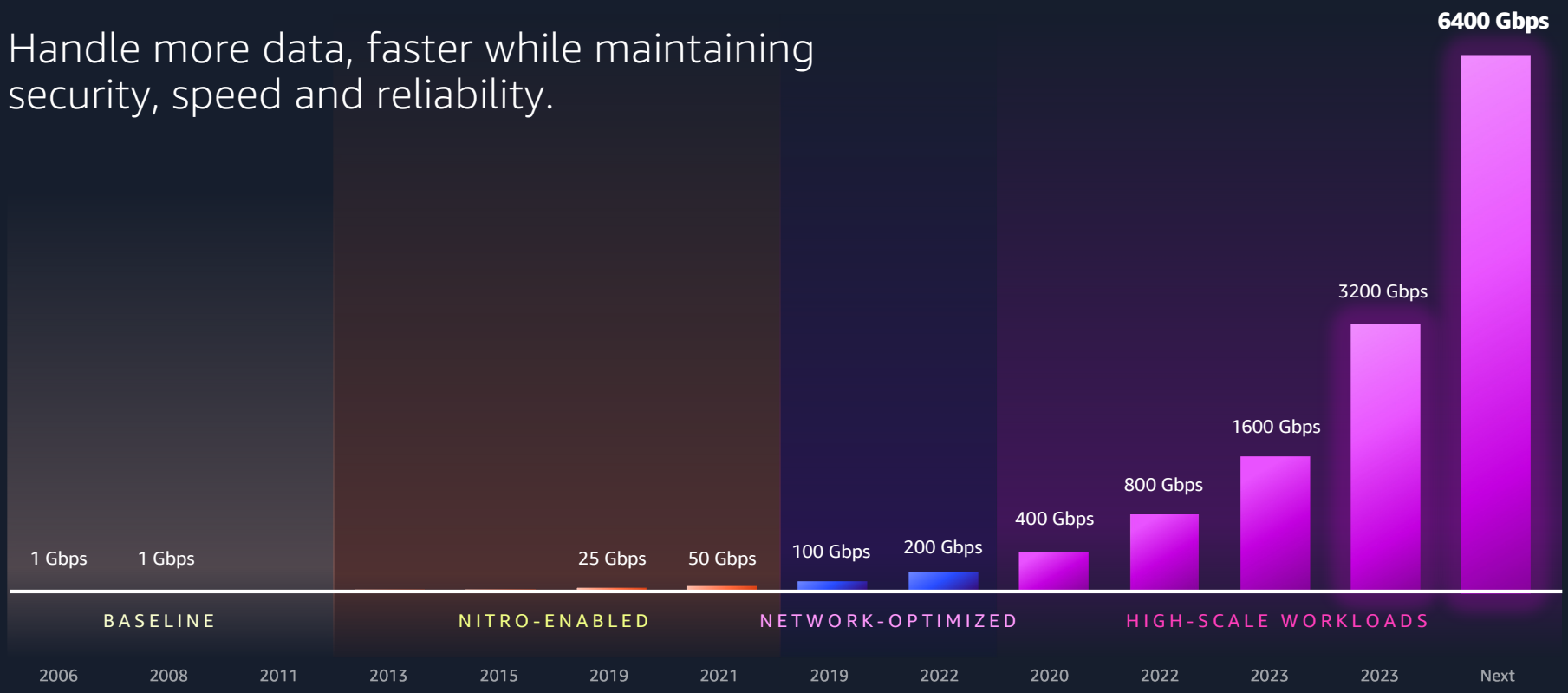
Up to 400Gbps
bandwidth
RDMA / NIC

MPI and NCCL
support



Network bandwidth

Handle more data, faster while maintaining security, speed and reliability.



EFA Generational Improvements

Generation	Latency	Bandwidth	New Features
1st (2018)	14 µs	100 Gbps	Send/recv semantics
2nd (2020)	9.3 µs	170 Gbps	RDMA semantics
3rd (2022)	6.8 µs	200 Gbps	ML hardware optimizations
4th (2024)	5.8 µs	400 Gbps	GB200 Support

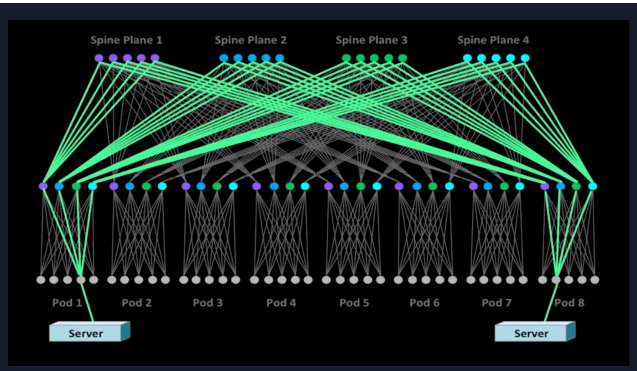
4th Generation Updates:

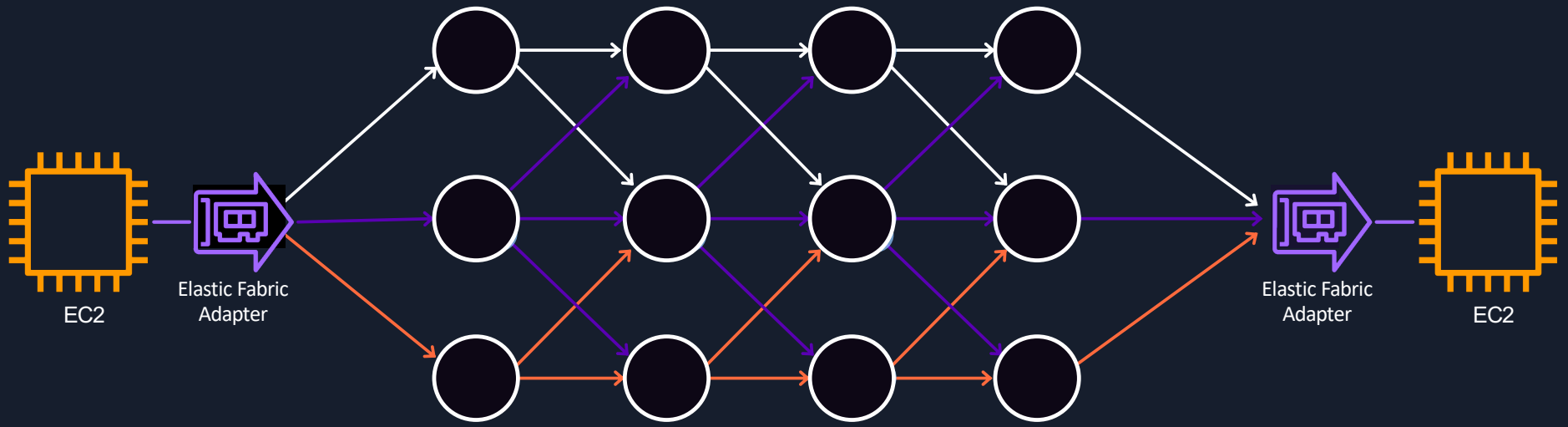
- Latency reduction to 5.2 µs in flight, also reduce average under load from 21 µs to 15 µs
- 768 Gbps bidirectional (96% peak), compared to ~95% on 3rd Generation and 92% on 2nd Generation

Captured from High-Performance Computing on AWS

SRD based ECMP

Providing consistent low latency for applications.





GPU Training Time By EFA/ENI (world size 4)

Llama-3-8B training with Huggingface + DeepSpeed Stage 3 **within Single Node**

DeepSpeed Config

```
{
  "fp16": { "enabled": false },
  "bf16": { "enabled": true },
  "zero_optimization": {
    "stage": 3,
    "offload_optimizer": { "device": "cpu", "pin_memory": true },
    "offload_param": { "device": "cpu", "pin_memory": true },
    "overlap_comm": true,
    "contiguous_gradients": true,
    "sub_group_size": 1e8,
    "reduce_bucket_size": 5e7,
    "stage3_prefetch_bucket_size": 5e7,
    "stage3_param_persistence_threshold": "auto",
    "stage3_max_live_parameters": 1e8,
    "stage3_max_reuse_distance": 1e8,
    "stage3_gather_16bit_weights_on_model_save": true
  },
  "gradient_clipping": "auto",
  "steps_per_print": 10,
  "train_micro_batch_size_per_gpu": 1,
  "gradient_accumulation_steps": 4,
  "wall_clock_breakdown": false
}
```

Trainer Config

```
per_device_train_batch_size=1,
gradient_accumulation_steps=4,
learning_rate=2e-5,
max_steps=50,
num_train_epochs=1,
bf16=True,
logging_steps=5,
deepspeed="llama-3-8b-stage3.json",
save_strategy="no",
gradient_checkpointing=True,
```

******* Running training *******

```
Num examples = 36,718
Num Epochs = 1
Instantaneous batch size per device = 1
Total train batch size (w. parallel, distributed & accumulation) = 16
Gradient Accumulation steps = 4
Total optimization steps = 50
Number of trainable parameters = 8,030,261,248
```

EFA / ENI
400 Gbps = 50GB/s

4 Pod [1 GPU / 1 EFA]
via NET/Libfabric/0/GDRDMA

39 min 32 sec

EKS kubeflow

4 Pod [1 GPU / 1 ENI]
via NET/Socket/0

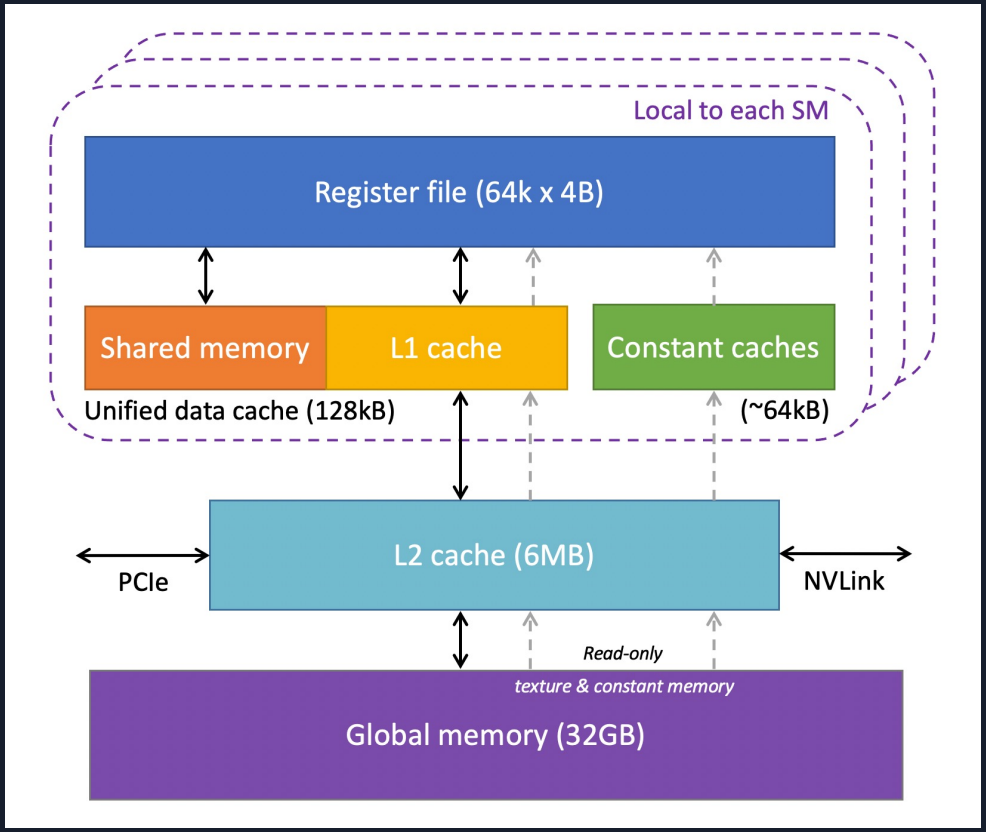
1 h 20 min 29 sec

EKS kubeflow

p4d.24xlarge (A100)

https://github.com/gnosia93/training-on-eks

GPU Memory Architecture



Register/Shared Memory:
closest to core and very fast and small

L2 Cache:
resides inside the GPU chip and fast usually tens of MB

Global Memory (Device Memory):
store parameter, gradient, optimizer status

Memory Bandwidth is very important for LLM and more important than computational performance.

GPU Training Time Comparison (world size 1)

Llama-3.2-1B training with Huggingface + DeepSpeed Stage 1 **with No offload optimizer**

DeepSpeed Config

```
{
  "train_micro_batch_size_per_gpu": "auto",
  "gradient_accumulation_steps": "auto",
  "zero_optimization": {
    "stage": 1,
    "offload_optimizer": {
      "device": "none"
    }
  },
  "bf16": {
    "enabled": true
  },
  "steps_per_print": 10
}
```

Trainer Config

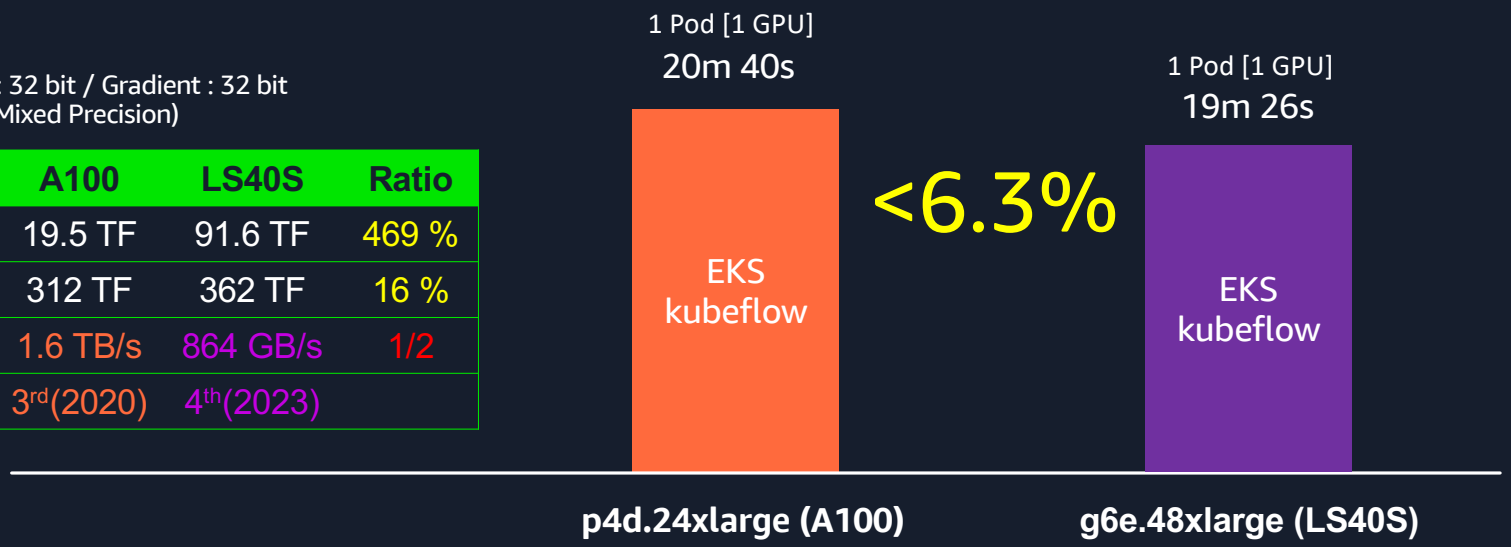
```
per_device_train_batch_size=16,
gradient_accumulation_steps=1,
learning_rate=2e-5,
num_train_epochs=1,
bf16=True,
logging_steps=5,
deepspeed="llama-3-1b-stage1.json",
save_strategy="no",
gradient_checkpointing=True,
```

******* Running training *******

```
Num examples = 36,718
Num Epochs = 1
Instantaneous batch size per device = 16
Total train batch size (w. parallel, distributed & accumulation) = 16
Gradient Accumulation steps = 1
Total optimization steps = 2,295
Number of trainable parameters = 1,235,814,400
```

Optimizer Stats : 32 bit / Gradient : 32 bit
Weight : 16 bit (Mixed Precision)

	A100	LS40S	Ratio
FP32	19.5 TF	91.6 TF	469 %
FP16	312 TF	362 TF	16 %
Mem B/W	1.6 TB/s	864 GB/s	1/2
Gen	3rd(2020)	4th(2023)	



GPU Memory Offloading Comparison (world size 1)

Llama-3.2-1B training with Huggingface + DeepSpeed Stage 1

DeepSpeed Config

```
{
    "train_micro_batch_size_per_gpu": "auto",
    "gradient_accumulation_steps": "auto",
    "zero_optimization": {
        "stage": 1,
        "offload_optimizer": {
            "device": "none"
        }
    },
    "bf16": {
        "enabled": true
    },
    "steps_per_print": 10
}
```

Trainer Config

```
per_device_train_batch_size=16,
gradient_accumulation_steps=1,
learning_rate=2e-5,
num_train_epochs=1,
bf16=True,
logging_steps=5,
deepspeed="llama-3-1b-stage1.json",
save_strategy="no",
gradient_checkpointing=True,
```

******* Running training *******

```
Num examples = 36,718
Num Epochs = 1
Instantaneous batch size per device = 16
Total train batch size (w. parallel, distributed & accumulation) = 16
Gradient Accumulation steps = 1
Total optimization steps = 2,295
Number of trainable parameters = 1,235,814,400
```

AWS storage portfolio

High throughput storage for AI dataloading and checkpointing

OBJECT

FILE

BLOCK

Amazon S3

Amazon EFS

Amazon FSx for Windows File Server

Amazon FSx for NetApp ONTAP

Amazon FSx for Lustre

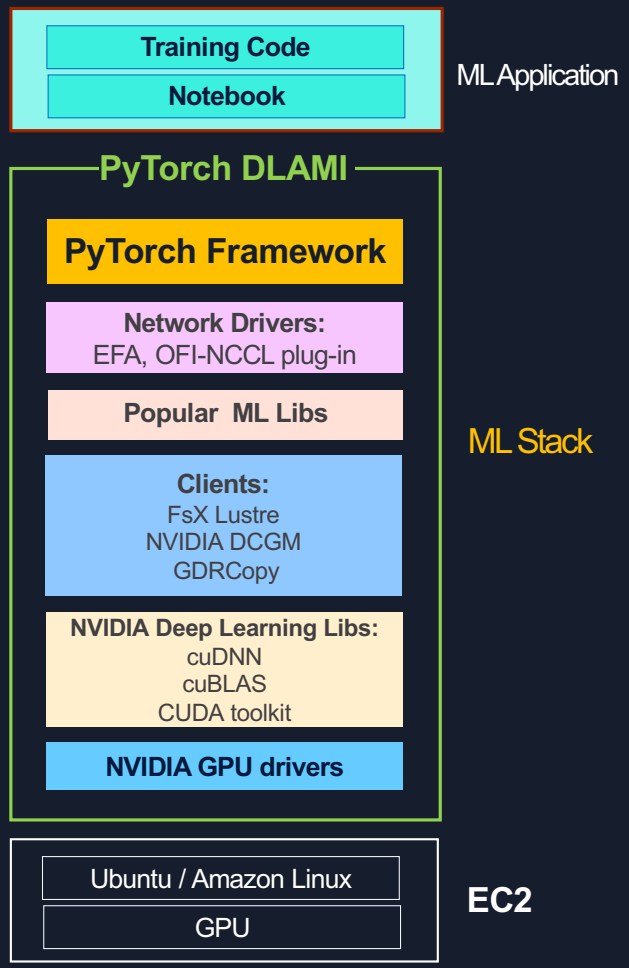
Amazon FSx for OpenZFS

Amazon EBS

https://github.com/awslabs/s3-connector-for-pytorch

Performance & Capacity

Service	FSx ONTAP	FSx OpenZFS	FSx Lustre	EFS	EBS	S3 Express One Zone
Latency	<1 ms	<0.5 ms	<1 ms	<1 ms	<1 ms	single-digit ms
Max. throughput	72 GB/s	21 GB/s	1000 GB/s	60 GB/s	4 GB/s	Tb/s
Max. IOPS	Millions	1-2 Million	Millions	2.5 Million	Hundreds of thousands	TPS Hundreds of Thousands
Maximum file system/vol size	Virtually unlimited (10s of PBs)	512 TiB	Multiple PBs	Virtually unlimited (10s of PBs)	64 TiB	Virtually unlimited (object)
Multi-AZ Options	Yes	Yes	No	Yes	No	No



AWS Deep Learning AMIs

Challenges in building the ML stack

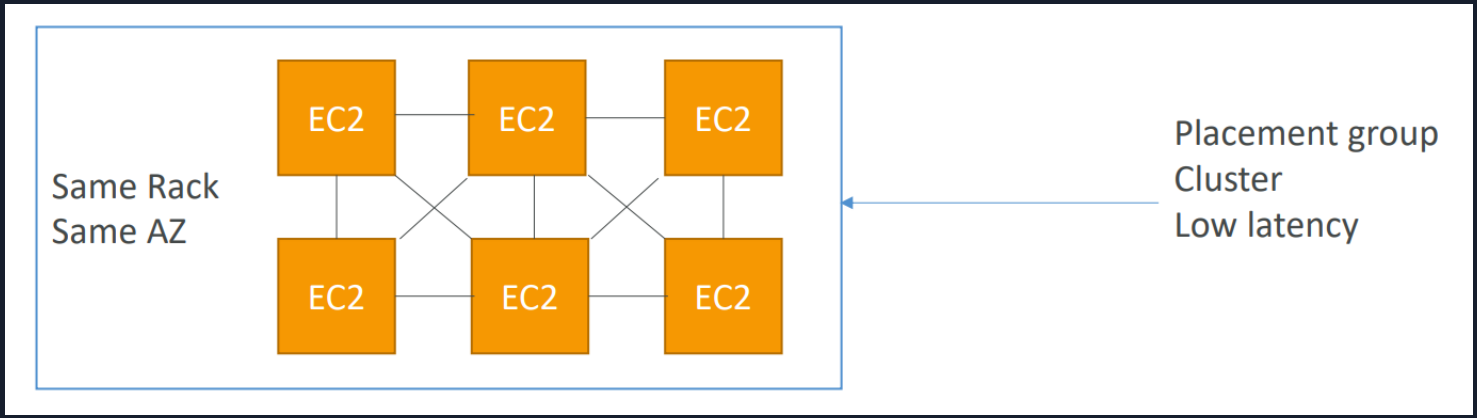
- ***Seamlessly integrating components***
- ***Maintaining currency***
- ***Security patching and testing***
- ***Complex architectures of new accelerators e.g. P5***
- ***Consumes expensive enterprise resources***

Packaging the ML stack into DLAMIs and DLCs that:

- ***work out-of-the-box to automatically setup ML environment***
- ***ensure stable inter-node connectivity for distributed training***
- ***rapidly scale the ML infrastructure, and***
- ***speed up training through the latest performant ML stack***

EC2 Placement Groups -Cluster

- A logical grouping of instances within **single Availability zone**
- Ideal for distributed applications that require low latency like **AI and HPC**



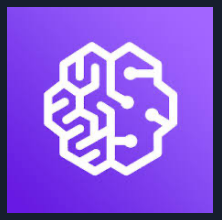
Distributed Training Infrastructure

AWS ParallelCluster

- SLURM On EC2
- Self Managed

Amazon EKS

- Kubeflow On EKS
- Slurm On EKS (Slinky)
- Self Managed



Sagemaker HyperPod

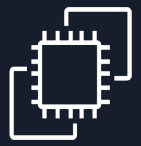
- HyperPod EKS
- HyperPod Slurm
- AWS Managed

AWS ParallelCluster

AWS ParallelCluster is an open source cluster management tool that makes it easy for you to deploy and manage High Performance Computing (HPC) clusters on AWS.

Integrated with AWS services you need

Amazon FSx for Lustre



Amazon EC2 instances

EFA

Data Preprocessing

- AWS Glue
- Amazon EMR
- EKS

Sagemaker AI

- Preprocessing Jobs
- spark / scikit learn …

- RAY [Data] on EC2
- Native Python

AWS Graviton Generations

Graviton Generation	Graviton1	Graviton2	Graviton3	Graviton4	Graviton5
Year	2018	2019	2021	2023	2025
Cores	16	64	64	96	192
Transistors	5 Billion	30 Billion	55 Billion	73 Billion	172 Billion

Why AWS Graviton?

Best price-performance in Amazon EC2

Advanced Arm cores - Neoverse N1 / V1~V3
Up to 20% cheaper than comparable X86 instances
+ Delivers up to 20% higher performance compared to X86 in specific workload

Energy efficient processors

Leading process nodes
End-to-end optimization

> 90,000 AWS customers

From startups to enterprises
In all 38 AWS regions across six-continents and over 300 Graviton-powered instance types

AWS Graviton4

for Data Preprocessing and Cluster Observability

Provisioning options for GPU instances

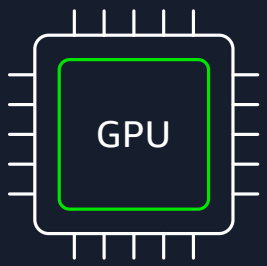
On-Demand Capacity Reservations (ODCR)

Spot Instances

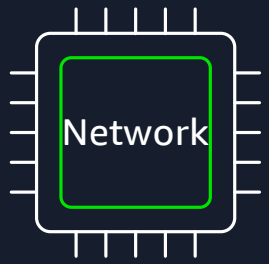
Capacity Blocks for ML

AI Cluster Primary Considerations (Wrap Up)

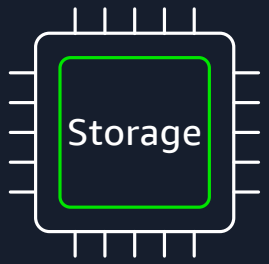
For distributed training of LLM models



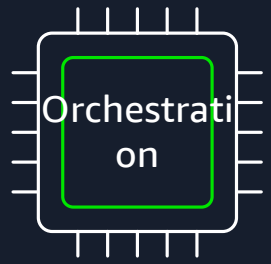
High speed interconnect
High Memory Bandwidth
High Memory Capacity
High Flops



RDMA / GPU Direct RDMA
High Bandwidth and Low latency scalable network
Lossless Network or Near-lossless Network



Distributed Parallel Storage
High Throughtput / Low latency
Backup / Recovery
Tiering



Gang Scheduling
Node Problem Detection
Node Auto Recovery
Job Auto Recovery
Observability
Container Support

+ Decoupling GPU instances from data preprocessing
+ Offload data preprocessing to CPUs
+ Topology aware parallelism

Resources

- Training On EKS - https://github.com/gnosia93/training-on-eks
- Ray[data] On EC2 - https://github.com/gnosia93/ray-on-ec2
- AWS Parallel Cluster - https://workshops.aws/categories/Compute?tag=ParallelCluster
- Sagemaker HyperPod - https://workshops.aws/categories/AI%2FML?tag=EC2

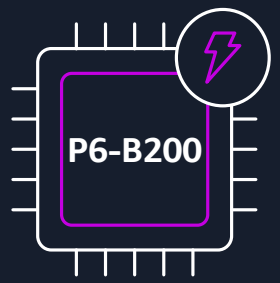
Thank you!

Amazon EC2 P-Type Instances

	Ampere	Hopper			Blackwell		
	P4d/p4de	P5*	P5e	P5en	P6-B200	P6-B300	P6e-GB200
NVIDIA GPU	A100 *8	H100 *8	H200 *8	H200 *8	B200 *8	B300 *8	B200 *72
GPU Memory	320GB HBM2/ 640GB HBM2e	640GB HBM3	1,128GB HBM3	1,128GB HBM3	1,440GB HBM3e	2,144GB HBM3e	13.4TB HBM3e
GPU Memory bandwidth	1.6 TB/s / 2TB/s	3.35 TB/s	4.8 TB/s	4.8 TB/s	7.7 TB/sec	7.7 TB/sec	7.7 TB/sec
CPU	Intel Cascade Lake	AMD EPYC 7R13	AMD EPYC 7R13	Intel Sapphire Rapids	Intel Emerald Rapid	Intel Emerald Rapid	NVIDIA Grace CPU (ARM 기반)
Network Bandwidth	400Gbps (EFA/ENA)	3.2 Tbps (EFAv2)	3.2 Tbps (EFAv2)	3.2 Tbps (EFAv3)	3.2 Tbps (EFAv4)	6.4Tbps (EFAv5), 300 Gbps dedicated ENA throughput	28.8 Tbps (EFAv4)
Peer-to-peer GPU communication	600 GB/s	900 GB/s	900 GB/s	900 GB/s	1,800GB/s	1,800GB/s	1,800GB/s
EBS Bandwidth	19 Gbps	80 Gbps	80 Gbps	100 Gbps	100 Gbps	100 Gbps	1,080Gbps
Nitro Version	Nitro v3	Nitro v4	Nitro v4	Nitro v5	Nitro v6	Nitro v6	Nitro v5
이전 인스턴스 대비 주요 변경	-	A100 → H100 2x GPU 메모리, 2x CPU 성능, 8x 네트워크 대역폭	H100 → H200 1.7x GPU 메모리	CPU 변경, EFAv3, 1.25x EBS 대역폭 35% latency↓	H200 → B200 1.27x GPU 메모리, CPU 변경, 2x 성능	B200→B300, 1.5x GPU 메모리, 2x 네트워크 대역폭, 1.5x FP4 compute (dense)	B200 → GB200 수퍼칩 구조, B200 72대 탑재, EC2 인스턴스 최초 Liquid cooling 적용

Amazon EC2 P6-B200 instances

Next-Generation NVL8 instances based on Blackwell GPUs and Nitro v6



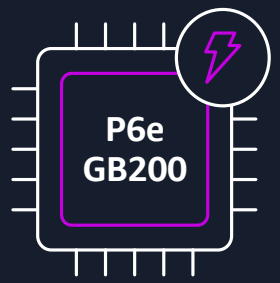
8x Blackwell B200 GPUs delivering 2x compute FLOPS and 25% higher memory at 60% higher memory bandwidth vs P5en to improve AI training and inference performance.

3.2 Tbps instance networking (400G/GPU) with fourth-generation Elastic Fabric Adapter (EFAv4) for optimized latency.

5th Generation Intel Xeon Scalable processor, 192 vCPUs, 2TiB system memory, and 30TB local SSD storage.

Amazon EC2 P6e-GB200 Ultaservers

Highest GPU performance for AI training and inference



Up to 72x Blackwell GPUs within one P6e-GB200 Ultraserver, providing 20x compute under NVLink vs P5en to train and inference the biggest models at the highest performance.

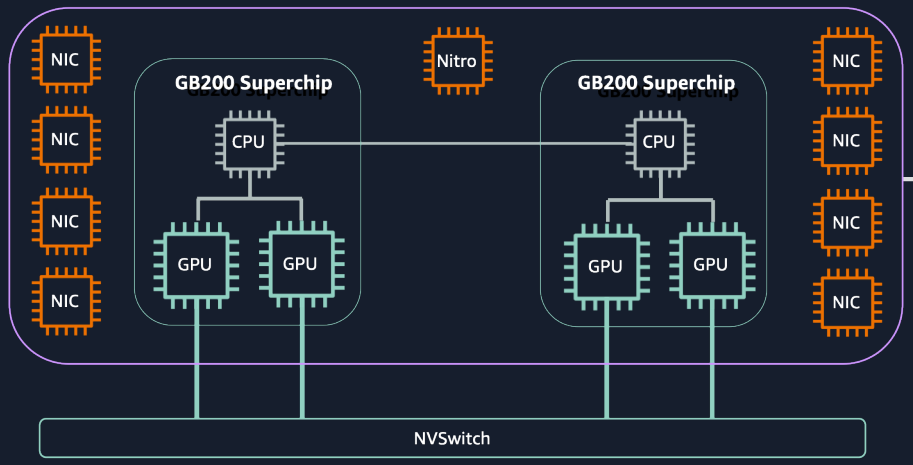
Up to 28.8 Tbps of instance networking (400G/GPU) with fourth-generation Elastic Fabric Adapter (EFAv4) for optimized latency.

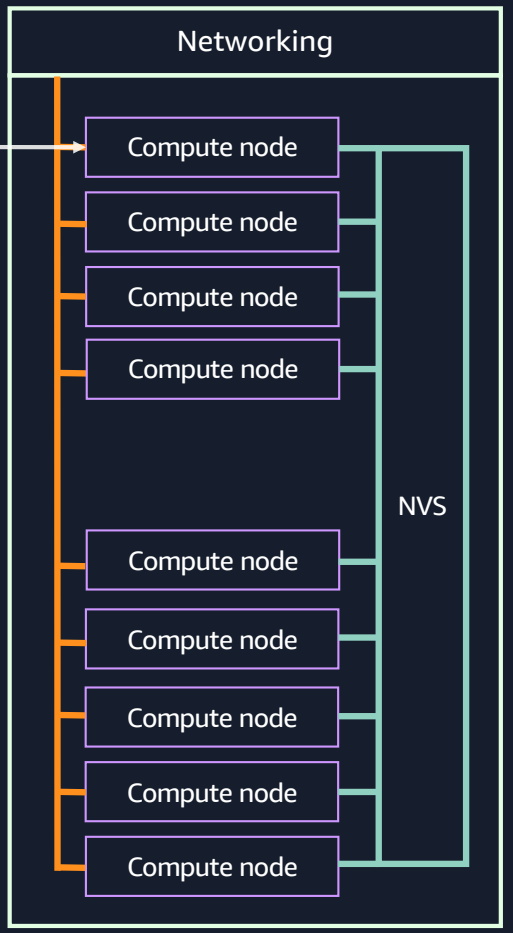
High-speed chip-to-chip (C2C) interconnect between Grace CPU and Blackwell GPU via superchip architecture.

First liquid-cooled platform at AWS, providing the highest GPU performance. Expected to GA in Q2’25.

P6e-GB200 Ultraserver Architecture

- New “Ultraserver” product with external NVLink across instances
- Up to 18x P6e-GB200 instances within one ultraserver
- 4x Blackwell GPUs per instance along with 400G/GPU instance networking, 22.5TB local storage, 960GB system memory

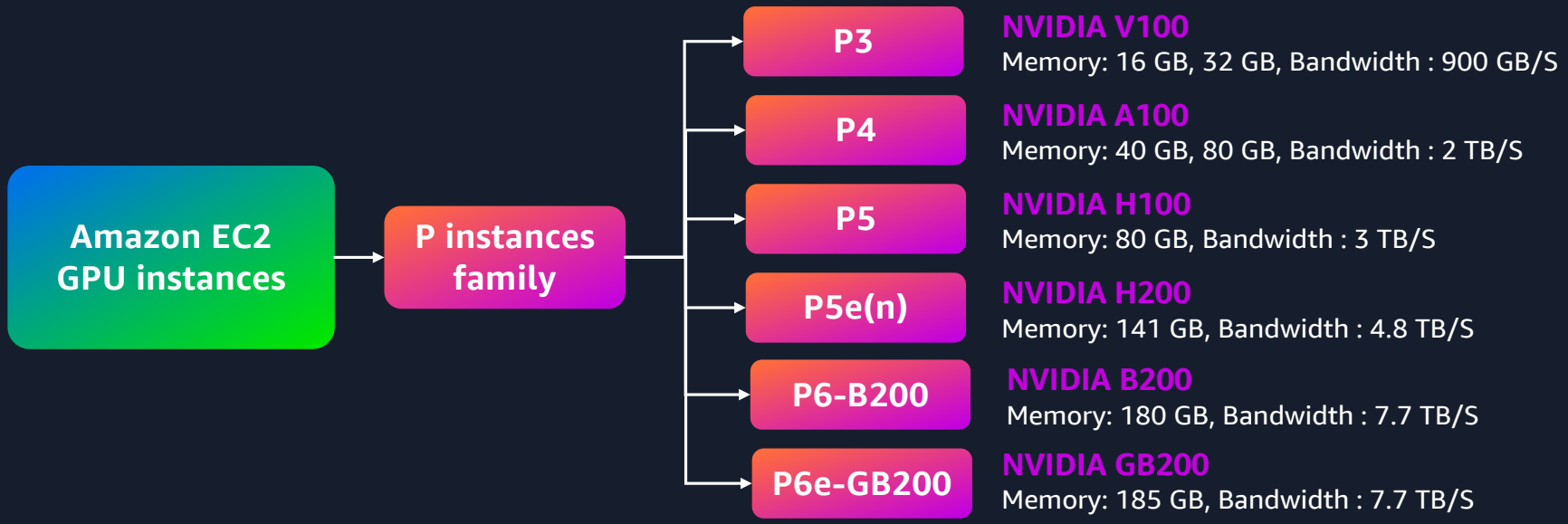




Amazon EC2 NVIDIA GPU instances

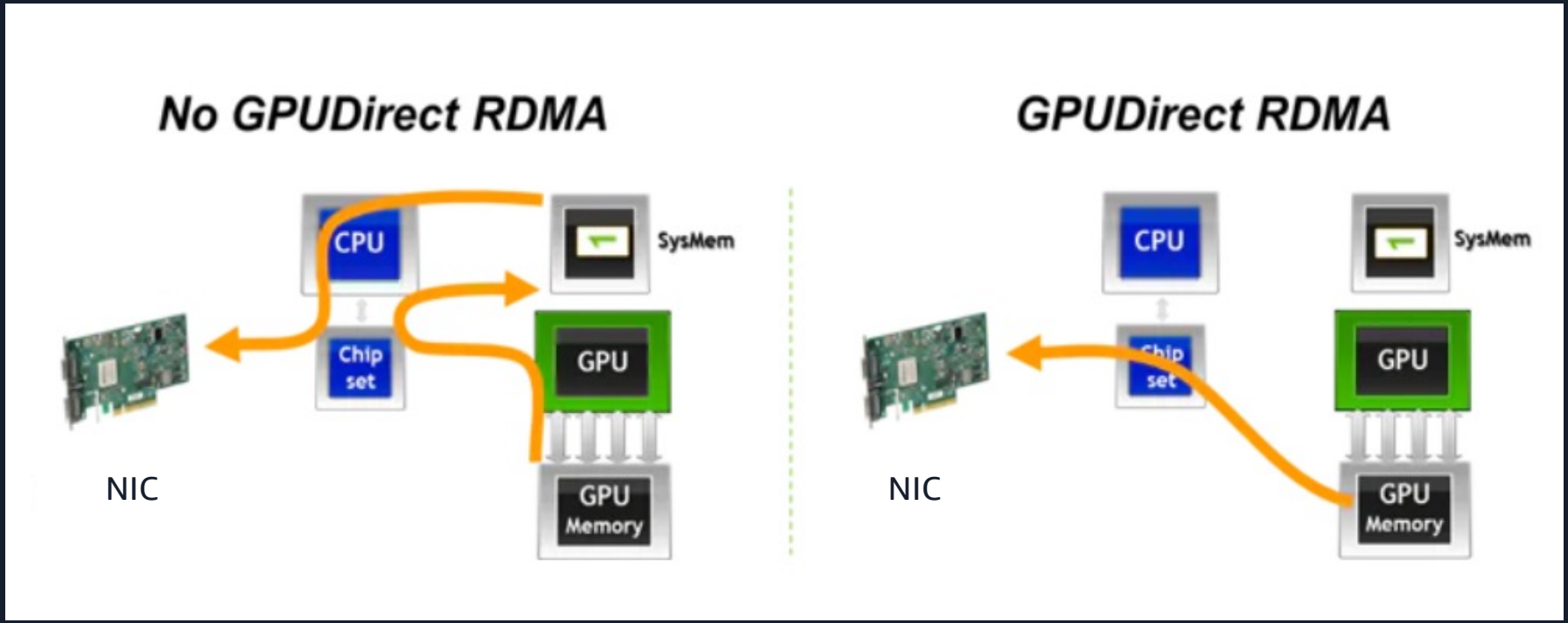
NVIDIA GPU architecture	Blackwell (2024)	Ada Lovelace (2022)	Hopper (2022)	Ampere (2020)	Turing (2018)	Volta (2017)	Pascal (2016)	Maxwell (2015)	Kepler (2012)
Amazon EC2 instances NVIDIA GPU	Amazon EC2 P6-B200 NVIDIA B200	Amazon EC2 G6 NVIDIA L4	Amazon EC2 P5 NVIDIA H100	Amazon EC2 P4 NVIDIA A100	Amazon EC2 G4 NVIDIA T4	Amazon EC2 P3 NVIDIA V100		Amazon EC2 G3 NVIDIA M60	Amazon EC2 P2 NVIDIA K80
	Amazon EC2 P6e-GB200 NVIDIA GB200	Amazon EC2 G6e NVIDIA L40S	Amazon EC2 P5e(n) NVIDIA H200	Amazon EC2 G5 NVIDIA A10G	Amazon EC2 G5g NVIDIA T4G				

Amazon EC2 NVIDIA GPU instances



Amazon EC2 NVIDIA GPU instances

RDMA vs GPU Direct RDMA



p4d.24xlarge NCCL Log with NVLink

```
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO === System : maxBw 240.0 totalBw 240.0 ===
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO CPU/0-0 (1/1/2)
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO + PCI[24.0] - GPU/0-101c0 (0)
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO               + NVL[240.0] - NVS/0-0
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO + PCI[24.0] - GPU/0-101d0 (1)
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO               + NVL[240.0] - NVS/0-0
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO + PCI[24.0] - GPU/0-201c0 (2)
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO               + NVL[240.0] - NVS/0-0
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO + PCI[24.0] - GPU/0-201d0 (3)
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO               + NVL[240.0] - NVS/0-0
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO + PCI[12.0] - NIC/0-101b0
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO + PCI[12.0] - NIC/0-201b0
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO + SYS[10.0] - CPU/0-1
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO CPU/0-1 (1/1/2)
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO + PCI[12.0] - NIC/0-901b0
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO + PCI[12.0] - NIC/0-a01b0
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO + SYS[10.0] - CPU/0-0
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO ==========================================
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO GPU/0-101c0 :GPU/0-101c0 (0/5000.0/LOC) GPU/0-101d0 (2/240.0/NVL) GPU/0-201c0 (2/240.0/NVL) GPU/0-201d0 (2/240.0/NVL) NVS/0-0 (1/240.0/NVL) CPU/0-0 (1/24.0/PHB) CPU/0-1 (2/10.0/SYS)
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO GPU/0-101d0 :GPU/0-101c0 (2/240.0/NVL) GPU/0-101d0 (0/5000.0/LOC) GPU/0-201c0 (2/240.0/NVL) GPU/0-201d0 (2/240.0/NVL) NVS/0-0 (1/240.0/NVL) CPU/0-0 (1/24.0/PHB) CPU/0-1 (2/10.0/SYS)
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO GPU/0-201c0 :GPU/0-101c0 (2/240.0/NVL) GPU/0-101d0 (2/240.0/NVL) GPU/0-201c0 (0/5000.0/LOC) GPU/0-201d0 (2/240.0/NVL) NVS/0-0 (1/240.0/NVL) CPU/0-0 (1/24.0/PHB) CPU/0-1 (2/10.0/SYS)
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO GPU/0-201d0 :GPU/0-101c0 (2/240.0/NVL) GPU/0-101d0 (2/240.0/NVL) GPU/0-201c0 (2/240.0/NVL) GPU/0-201d0 (0/5000.0/LOC) NVS/0-0 (1/240.0/NVL) CPU/0-0 (1/24.0/PHB) CPU/0-1 (2/10.0/SYS)
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO Setting affinity for GPU 0 to 0-23,48-71
llama-3-8b-node-0-0:212:1380 [0] NCCL INFO NVLS multicast support is not available on dev 0 (NVLS_NCHANNELS 0)
```

1. High-Speed Interconnect (NVLink)
The log confirms your system is using **NVLink (NVL)** via an **NVSwitch (NVS)**.
- **maxBw 240.0**: It detected a massive bandwidth of **240 GB/s** between GPUs.
- This indicates a high-end setup (like an AWS P4/P5 instance) rather than standard PCIe, which is much slower.

2. Topology Mapping
NCCL ranks the "distance" between components to prioritize speed:
- **NVL (NVLink)**: Top priority. The 4 GPUs are fully interconnected at 240 GB/s.
- **PHB (PCI Bridge)**: Secondary path used for CPU-to-GPU communication (24 GB/s).
- **SYS (System)**: The slowest path, involving cross-socket CPU communication (10 GB/s).

3. Optimization Steps
- **CPU Affinity**: Setting affinity for GPU 0 to 0-23,48-71 means NCCL pinned the GPU to specific CPU cores to prevent bottlenecks during data loading.
- **NIC Detection**: It identified 4 Network Interface Cards (**NICs**), which is critical for the **EFA (Elastic Fabric Adapter)** performance mentioned in your workshop syllabus.

4. Minor Warning
- **NVLS multicast support is not available**: This means the hardware or driver doesn't support NVLink Switch Multicast. It won't stop your training, but it means certain specialized collective optimizations are skipped.

Summary: Your environment is **perfectly configured** for high-performance LLM training. The GPUs are talking to each other at maximum speed via NVLink.

g6e.48xlarge NCCL Log with PCIe (PHB)

```
llama-3-8b-node-0-0:234:1819 [2] NCCL INFO NVLS multicast support is not available on dev 2 (NVLS_NCHANNELS 0)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO === System : maxBw 12.0 totalBw 12.0 ===
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO CPU/0-0 (1/2/-1)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO + PCI[24.0] - PCI/0-85000 (1d0f02001d0f0200)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO               + PCI[24.0] - NIC/0-9b000
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO               + PCI[24.0] - NIC/0-9c000
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO + SYS[16.0] - CPU/0-1
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO + PCI[12.0] - GPU/0-9e000 (0)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO + PCI[12.0] - GPU/0-a0000 (1)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO + PCI[12.0] - GPU/0-a2000 (2)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO + PCI[12.0] - GPU/0-a4000 (3)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO CPU/0-1 (1/2/-1)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO + PCI[24.0] - PCI/0-a6000 (1d0f02001d0f0200)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO               + PCI[24.0] - NIC/0-bc000
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO               + PCI[24.0] - NIC/0-bd000
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO + SYS[16.0] - CPU/0-0
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO ==========================================
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO GPU/0-9e000 :GPU/0-9e000 (0/5000.0/LOC) GPU/0-a0000 (2/12.0/PHB) GPU/0-a2000 (2/12.0/PHB) GPU/0-a4000 (2/12.0/PHB) CPU/0-0 (1/12.0/PHB) CPU/0-1 (2/12.0/SYS)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO GPU/0-a0000 :GPU/0-9e000 (2/12.0/PHB) GPU/0-a0000 (0/5000.0/LOC) GPU/0-a2000 (2/12.0/PHB) GPU/0-a4000 (2/12.0/PHB) CPU/0-0 (1/12.0/PHB) CPU/0-1 (2/12.0/SYS)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO GPU/0-a2000 :GPU/0-9e000 (2/12.0/PHB) GPU/0-a0000 (2/12.0/PHB) GPU/0-a2000 (0/5000.0/LOC) GPU/0-a4000 (2/12.0/PHB) CPU/0-0 (1/12.0/PHB) CPU/0-1 (2/12.0/SYS)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO GPU/0-a4000 :GPU/0-9e000 (2/12.0/PHB) GPU/0-a0000 (2/12.0/PHB) GPU/0-a2000 (2/12.0/PHB) GPU/0-a4000 (0/5000.0/LOC) CPU/0-0 (1/12.0/PHB) CPU/0-1 (2/12.0/SYS)
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO Setting affinity for GPU 0 to 0-47,96-143
llama-3-8b-node-0-0:232:1816 [0] NCCL INFO NVLS multicast support is not available on dev 0 (NVLS_NCHANNELS 0)
```

1. Hardware Inventory (4 GPUs detected)
The system has identified **4 GPUs** (indexed 0 to 3) and their respective memory addresses (e.g., 0-9e000). It also found 4 Network Interface Cards (**NICs**) for inter-node communication.

2. Bandwidth Bottleneck

- **maxBw 12.0 totalBw 12.0**: NCCL has capped the communication bandwidth at **12.0 GB/s**.
- **Interpretation**: This is relatively low for high-performance AI tasks. It suggests the GPUs are connected via **PCIe Gen3** or a limited interconnect rather than high-speed **NVLink** (which usually hits 25-50+ GB/s per link).

3. Connection Topology (How they talk)
The matrix at the bottom describes the "distance" between components:

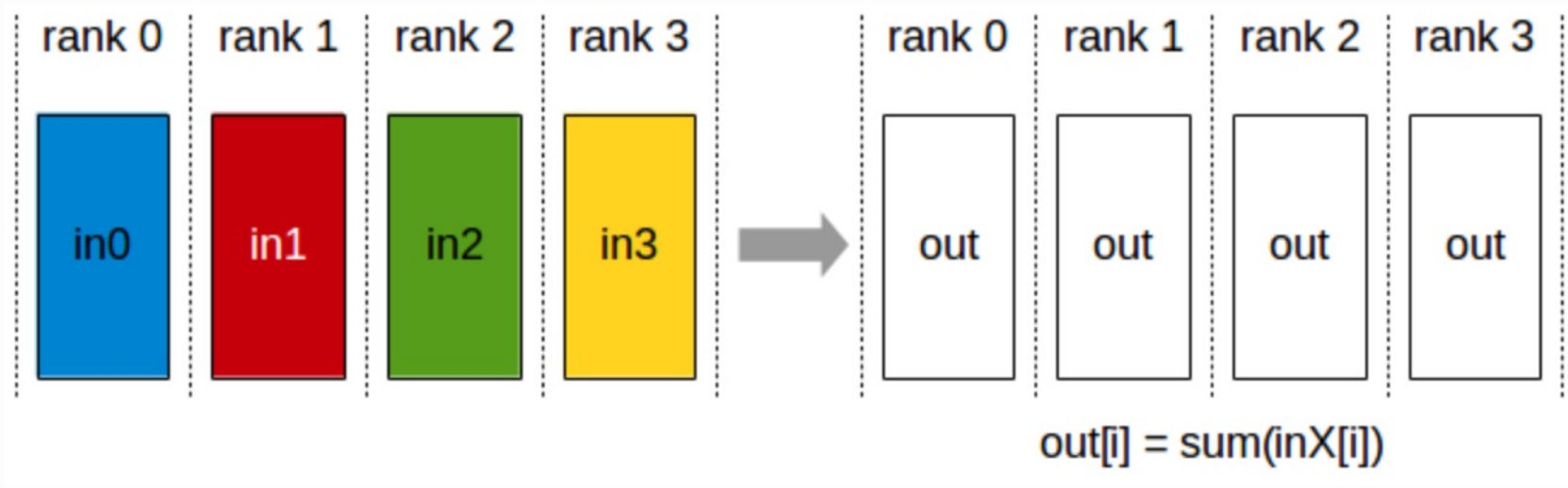
- **LOC (Local)**: The GPU talking to itself (5000.0 GB/s - internal memory speed).
- **PHB (PCH Bridge)**: GPUs are communicating via the **PCIe bus** through the CPU's host bridge. NVIDIA NCCL Documentation refers to this as a standard PCIe path.
- **SYS (System)**: Communication that must cross between different CPU sockets (NUMA nodes), which is the slowest path.

4. CPU Affinity

- **Setting affinity for GPU 3 to 0-47, 96-143**: NCCL is pinning the process for GPU 3 to specific CPU logical cores to reduce latency and prevent "context switching" during data transfers.

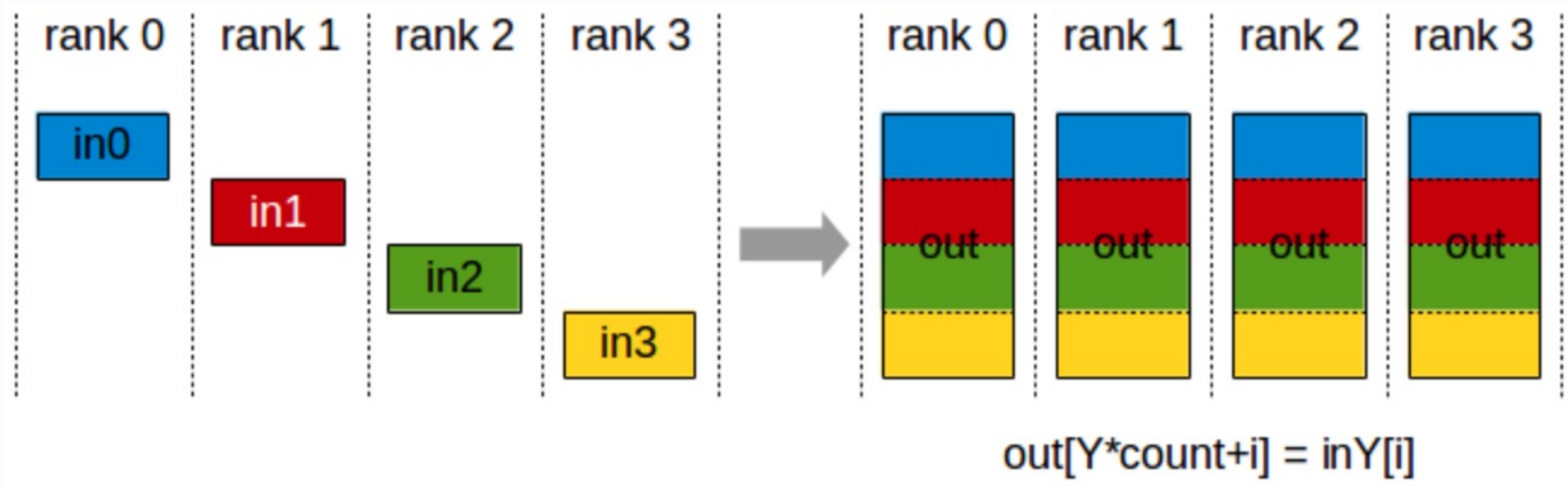
Summary
The system is **functioning correctly** and has finished mapping the hardware. However, because it is using **PCIe (PHB)** instead of **NVLink**, your multi-GPU communication (like AllReduce operations during training) will be limited to **12 GB/s**, which may become a bottleneck if you are doing heavy distributed training.

NCCL – All Reduce



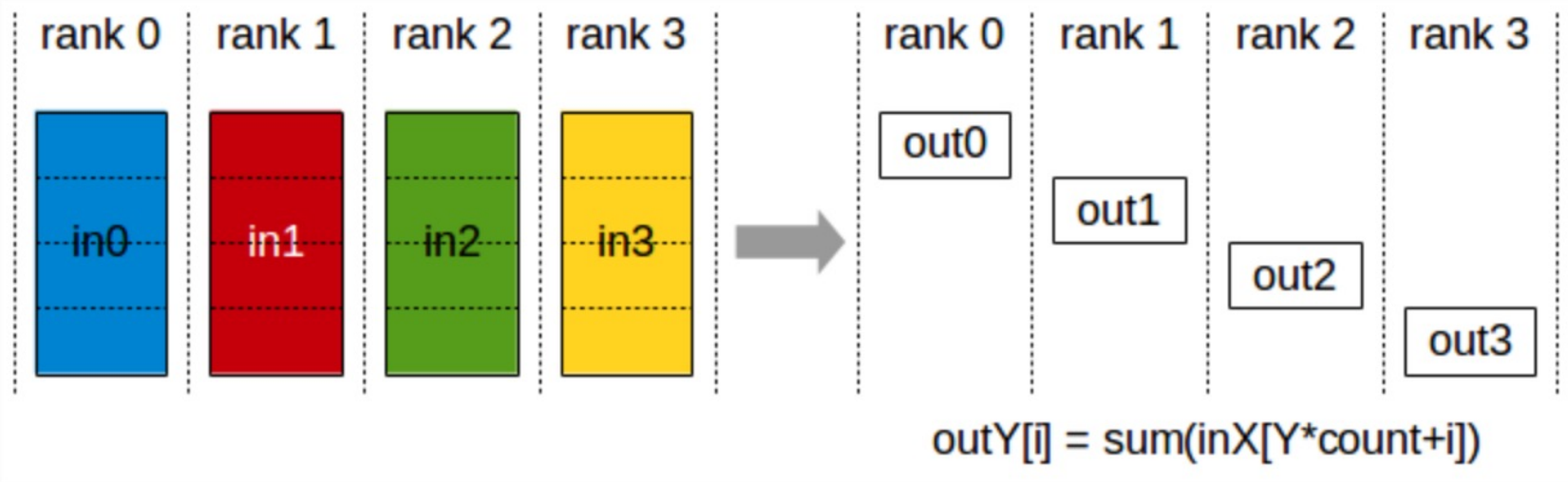
All-Reduce operation: each rank receives the reduction of input values across ranks.

NCCL – All Gather



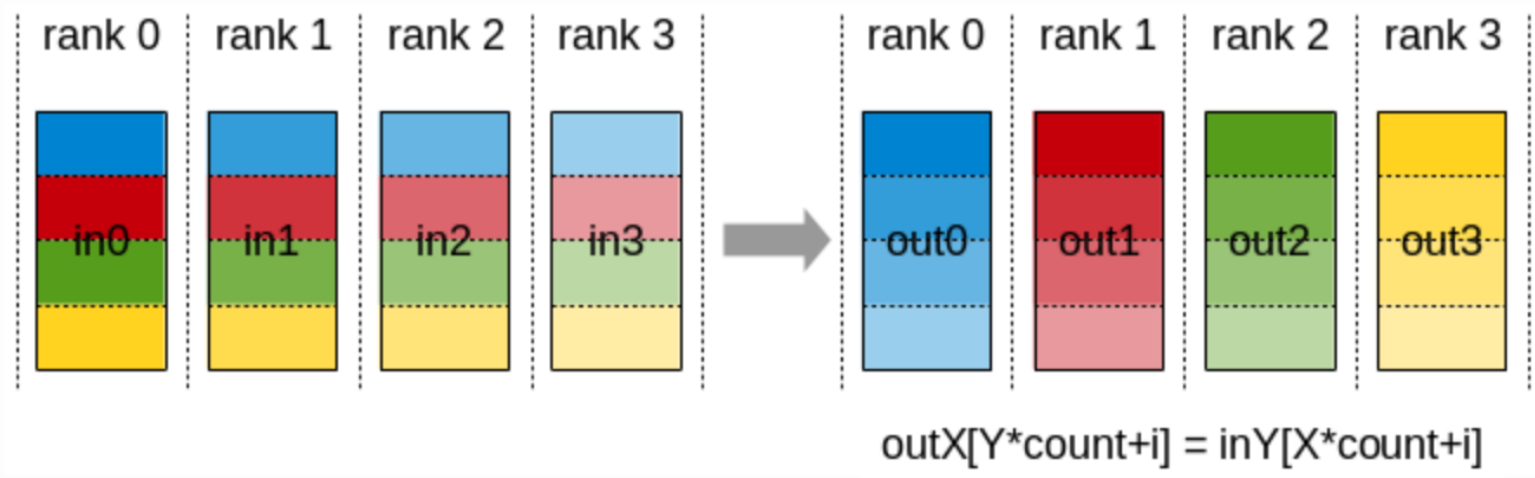
AllGather operation: each rank receives the aggregation of data from all ranks in the order of the ranks.

NCCL – Reduce / Scatter



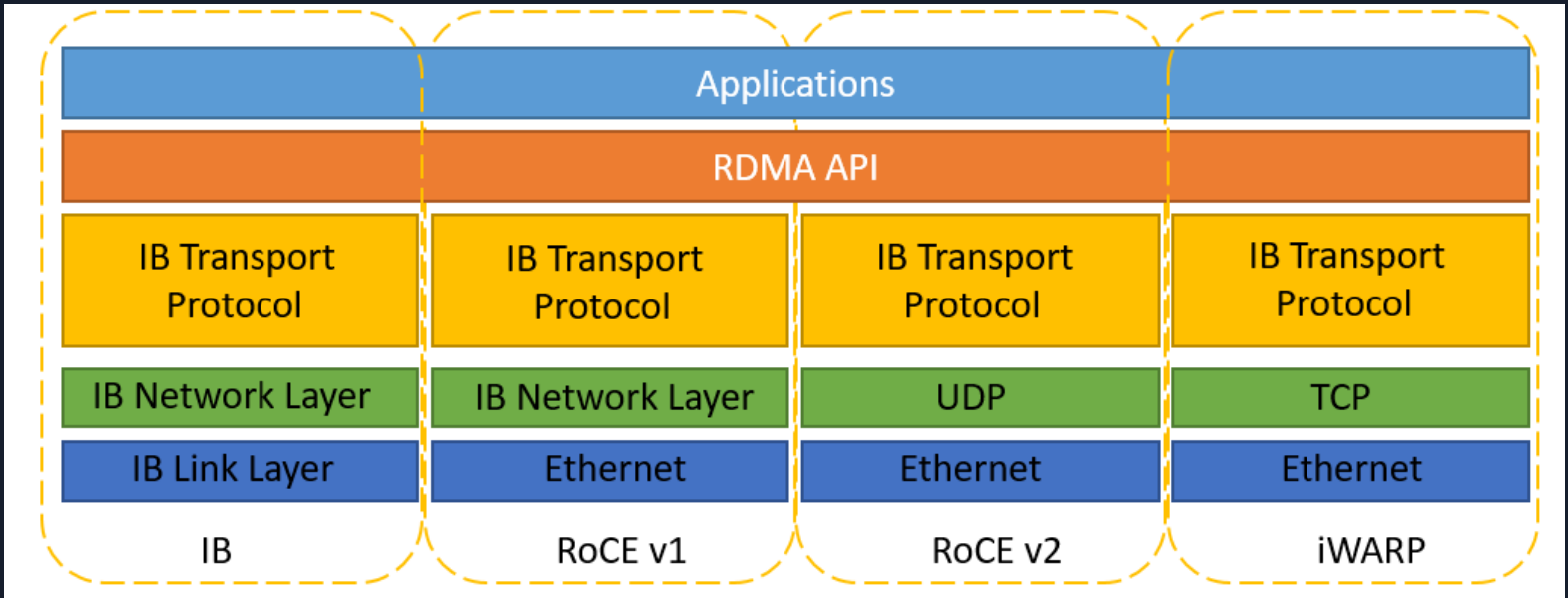
Reduce-Scatter operation: input values are reduced across ranks, with each rank receiving a subpart of the result.

NCCL – All to All



AlltoAll operation: exchanges data between all ranks, where each rank sends different data to every other rank and receives different data from every other rank.

RDMA over Converged Ethernet (RoCE)

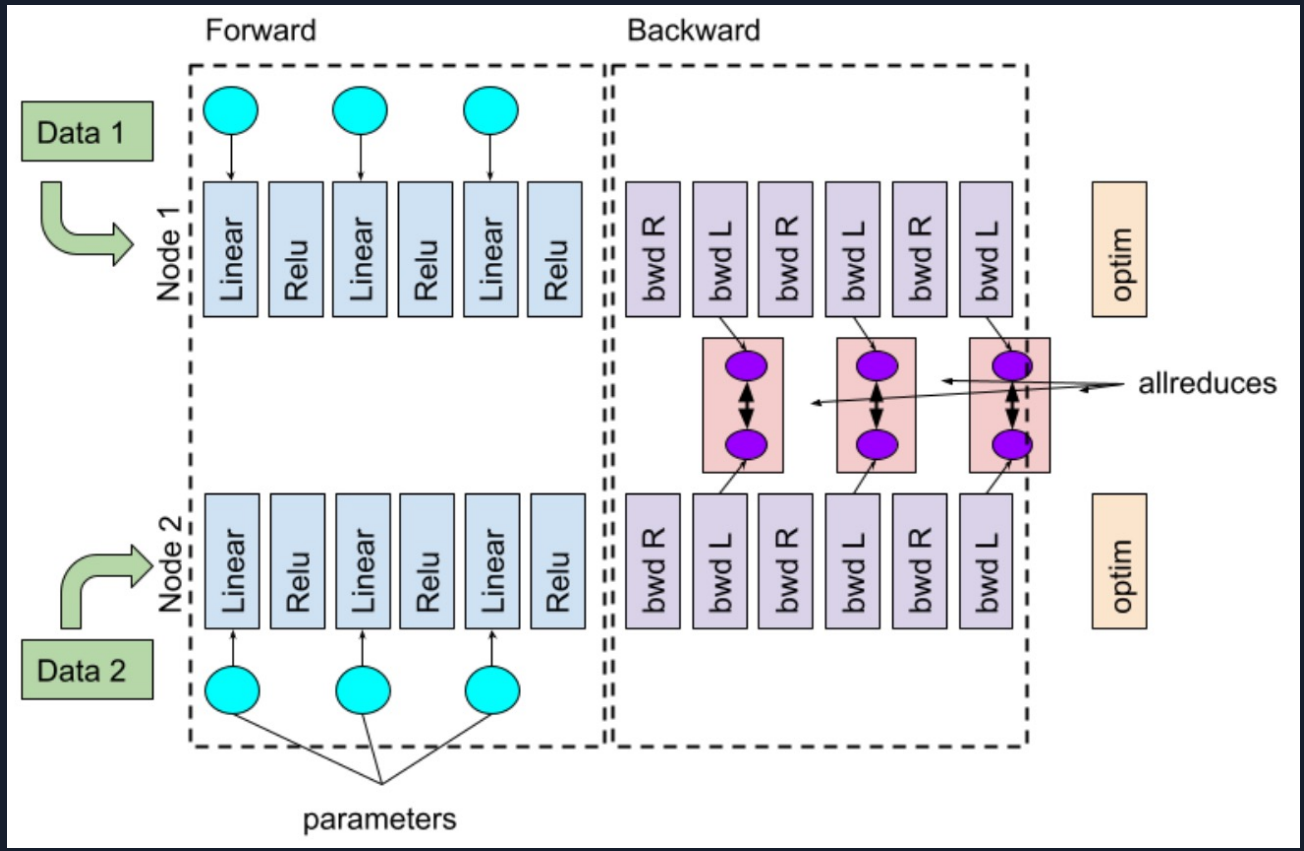


- RoCE required dedicated RoCE NIC and L2/L3 Switch supporting Data Center Bridging
- For lossless network over the Ethernet , network congestion is controled with PFC and ECN
- Typical latency of RoCE is between 1.5 µs ~ 10 µs / Switch Latency per hop is 0.5 µs

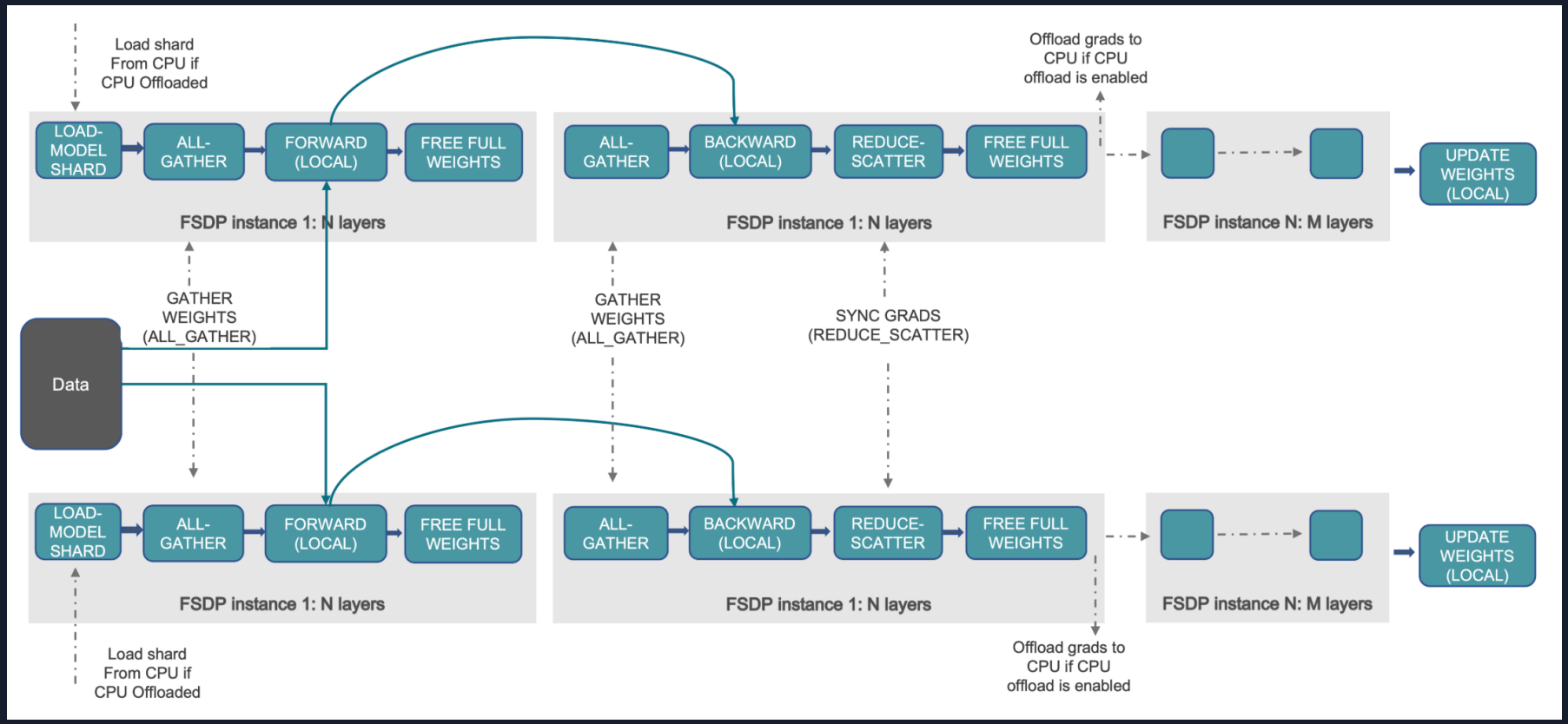
Latency and Bandwidth of Distributed Training

Parallelism Strategy	Most Critical Factor	Description
Data Parallelism	**Bandwidth**	Must sync huge gradient tensors every step.
FSDP/Tensor Parallelism	**Latency**	Frequent, synchronous communication within a single layer.
Pipeline Parallelism	**Latency**	Sensitive to latency for point-to-point "hops" between stages.

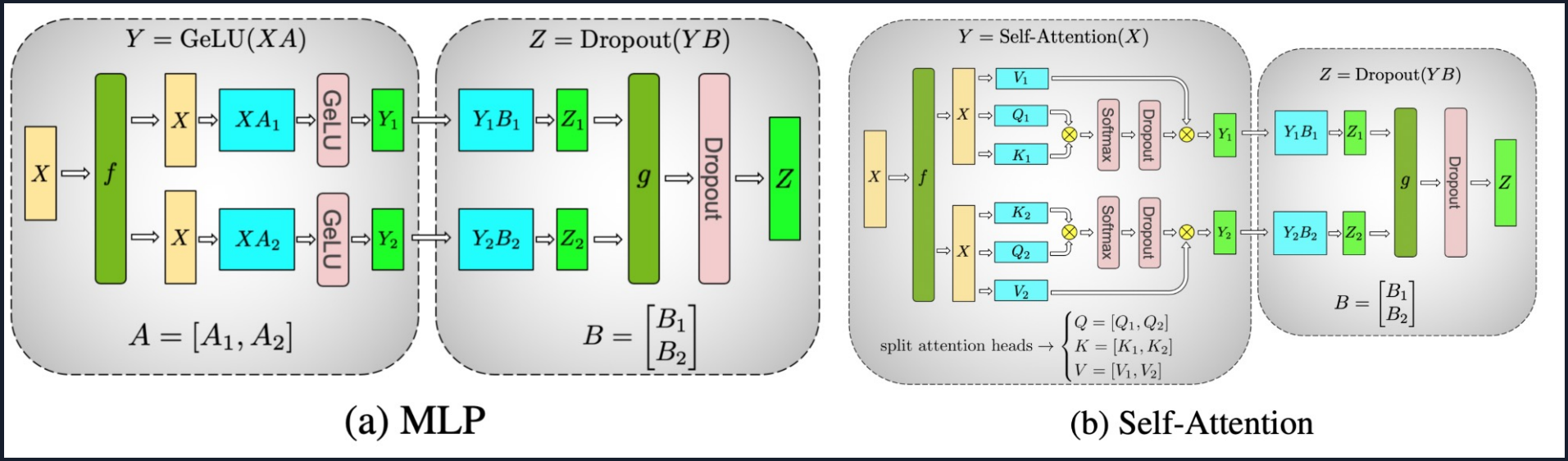
Distributed Data Parallel (DDP)



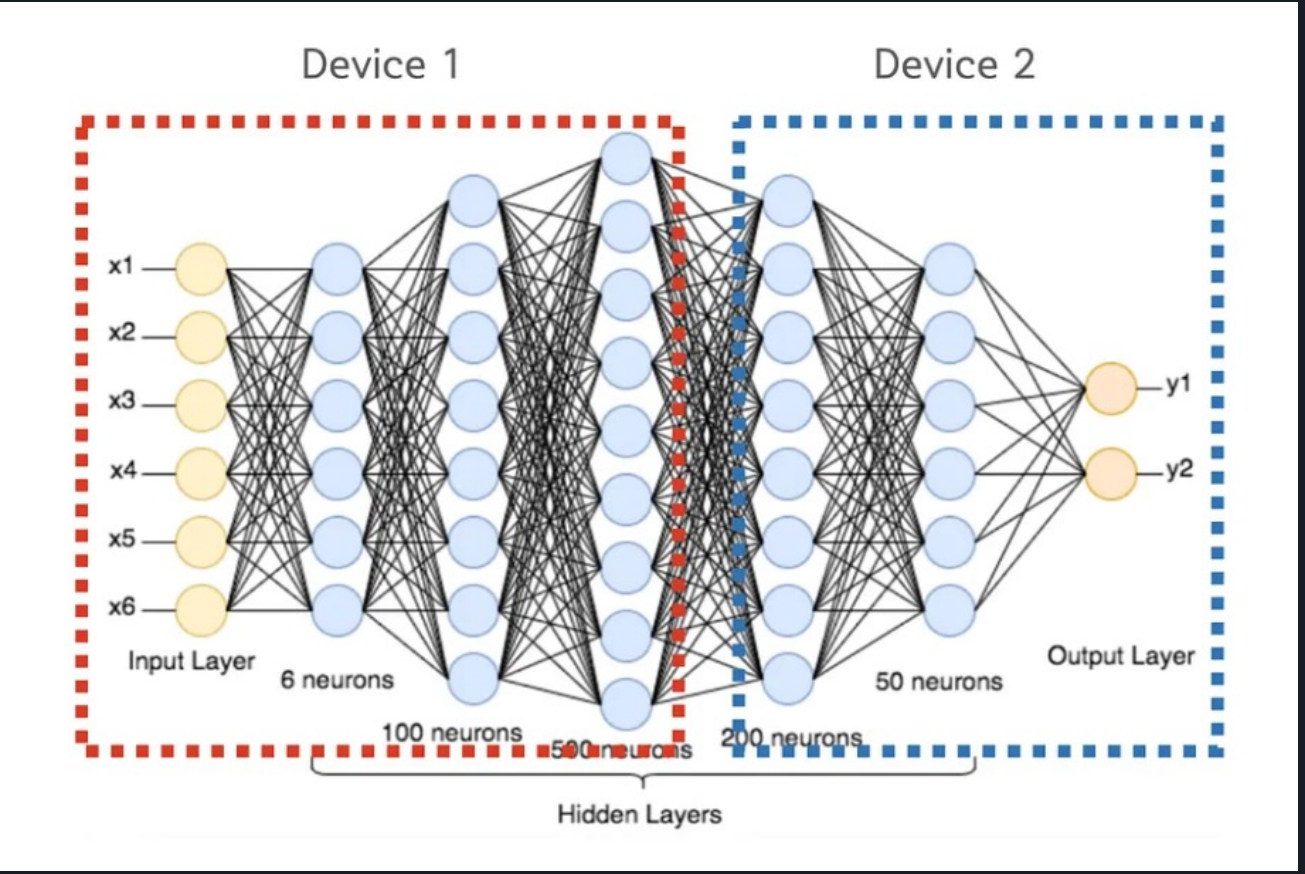
Fully Shared Data Parallel (FSDP)



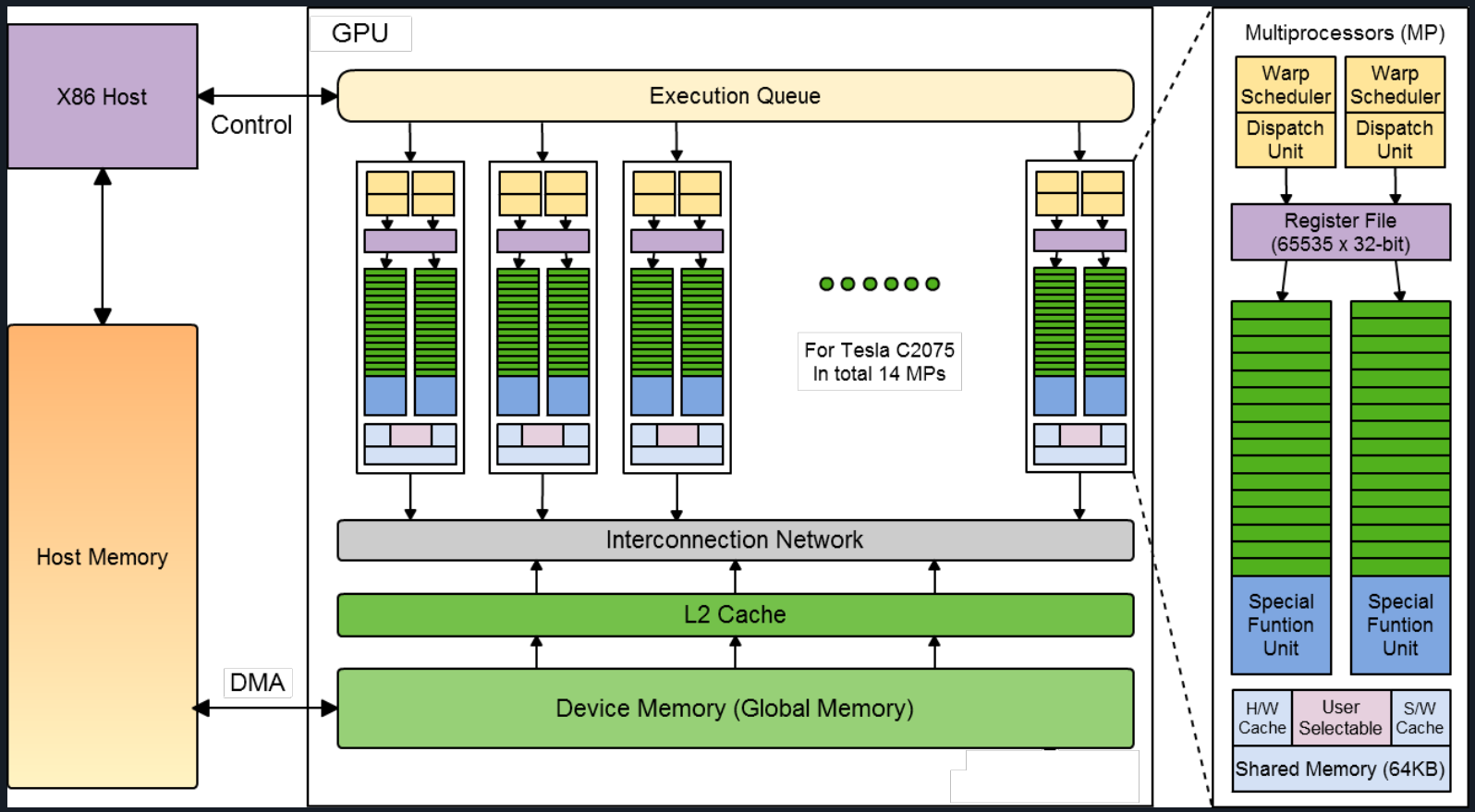
Tensor Parallel



Pipeline Parallel



GPU Architecture



How to choose right GPU for LLM ?

1. Calaculate total memory of target model for training
   - parameter, gradient, optimizer state, activation, master weights with Mixed precision
   - communication memory, CUDA context, temporary buffer and memory fragmentation

2. Check if single node GPU instance with 4 or 8 GPUs can occupy it ?
   - If so, choose GPU based on your objective, but usually GPU having NVLink/Switch is the best.

3. If you need multi-node GPU instances, choose bigger size memory GPU instances to reduce the number of training instances.

NVLink is 600GB/s ~
EFA is 50GB/s ~ per instance having 8 GPUs
GPU is always hugry, waiting data feed

LLM Memory Requirements

Weights (2bytes) + Gradient (4bytes) + Optimizer Stats (12bytes)

+ Activation
(batch size + sequence length)

- **Master Copy Weights**
- **Momentum**
- **Variance**

https://apxml.com/tools/vram-calculator

The Type of GPU Cores

CUDA Core

scala/vector ALU

Tensor Core

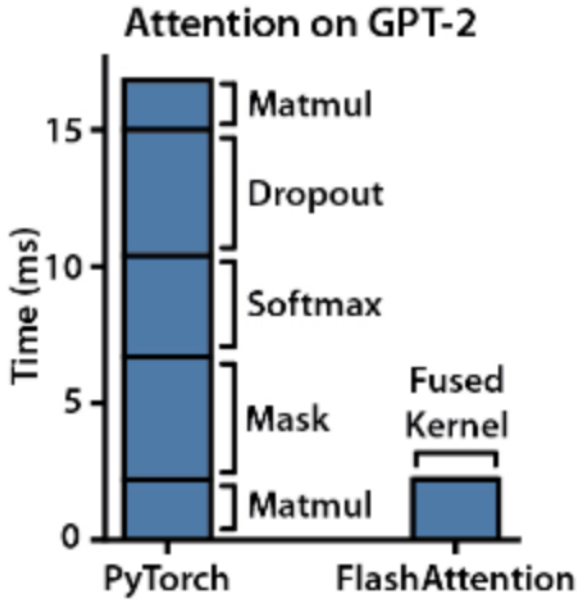
matrix ALU

4x4, 8x8, 16x16

RT Core

simulate light behaviour

Kernel Fusion



https://github.com/Dao-AILab/flash-attention

NVIDIA Transformer Engine (TE)

A library specifically designed to unlock the FP8 Tensor Cores from NVIDIA H100 GPU. Even if a developer codes in FP32 or BF16, TE acts as a bridge that automatically converts data into FP8 at the hardware level to maximize throughput. The Transformer Engine is a specialized optimization layer that fuses Software Intelligence (auto-scaling/casting) with H100 FP8 units to make AI training faster and leaner without the headache of manual tuning.

- On-the-fly Casting - data conversion (e.g., FP32 to FP8) in real-time within the optimized kernels just before the math happens.
- Numerical Stability (Delayed Scaling): To prevent precision loss (overflow/underflow), it tracks the maximum absolute values (amax) during execution and uses these statistics to automatically calculate and apply scale factors for the next steps.
- Achieves up to 9x faster training compared to A100 with new hardware-level instructions (ISA) for FP8.
- While it maintains "Master Weights" in FP32/BF16 for accuracy, it stores the massive Activation data in FP8. This significantly reduces the memory footprint, allowing for larger batch sizes or bigger models

https://github.com/NVIDIA/TransformerEngine

Flash Attention

FlashAttention is an IO-aware exact attention algorithm designed to speed up Transformer models and reduce memory footprint.

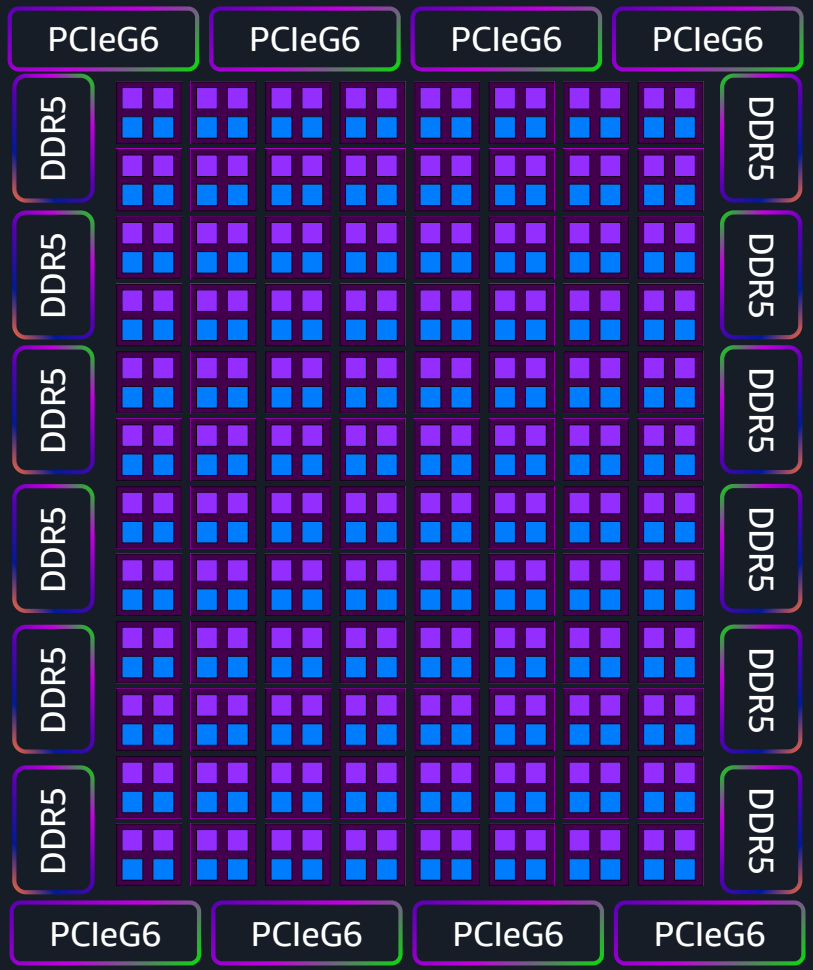
- Tiling: Breaking the attention matrix into blocks to fit into the fast SRAM (internal GPU memory).
- Recomputation: Avoiding the storage of massive intermediate attention matrices in HBM (main GPU memory), recalculating them on-the-fly instead.
- FlashAttention-1 & 2 (Fully Supported): These were specifically optimized for the Ampere architecture (A100). They provide a massive performance boost (2-4x speedup) on A100 GPUs compared to standard attention.
- FlashAttention-3 (Limited/Optimized for H100): Version 3 targets the Hopper architecture (H100) to utilize new hardware features like FP8 and TMA. While it may run on A100, you won't see the version 3 specific speed gains.

Graviton5

192 cores

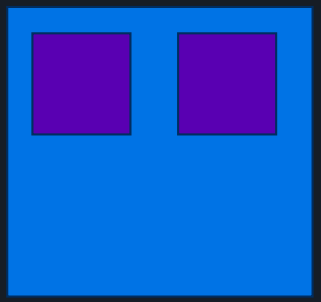
3nm semiconductor process

Up to 25% higher perf/core



CORES

Graviton5



Arm Neoverse V3 core

Bigger branch predictor / More advanced prefetchers

2MB L2 cache / 192MB L3 cache (5.3x of Graviton4)

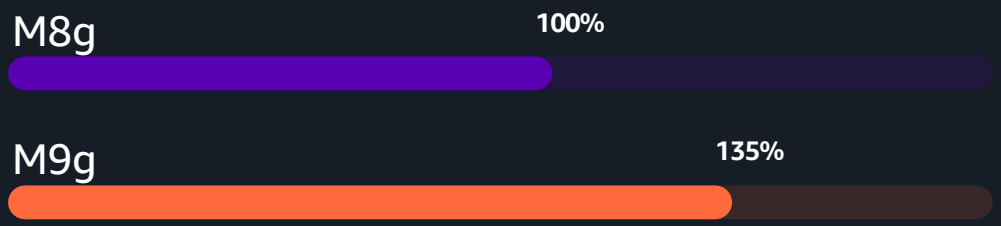
Support for up to DDR5-8800

<100ns DRAM access latency

First CPU with PCIe gen6 in AWS

512 GB/s of I/O connectivity

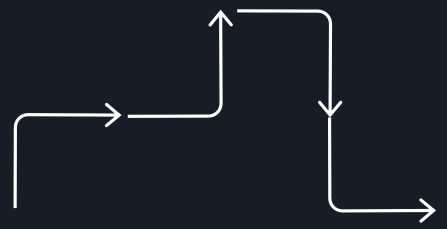
PyTorch

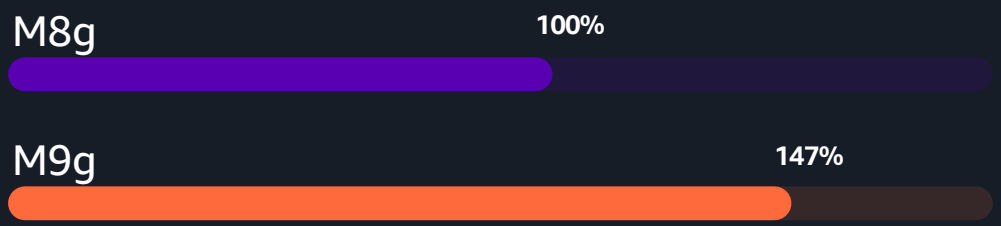


PyTorch Deep Learning Container running BERT Q&A, BERT and RoBERTa sentiment analysis, mask word prediction, resnet50, and Gemma-2-2b across oneDNN and OpenBLAS backends, and taking the Geomean of inferences/second.

Java Performance

Delivery





8vCPU instances under test running Internal Java workload