Hardware Acceleration for Deep Learning: Choosing the Right Compute for AI Models

Yared Leake, Tawer Kidanu

Introduction

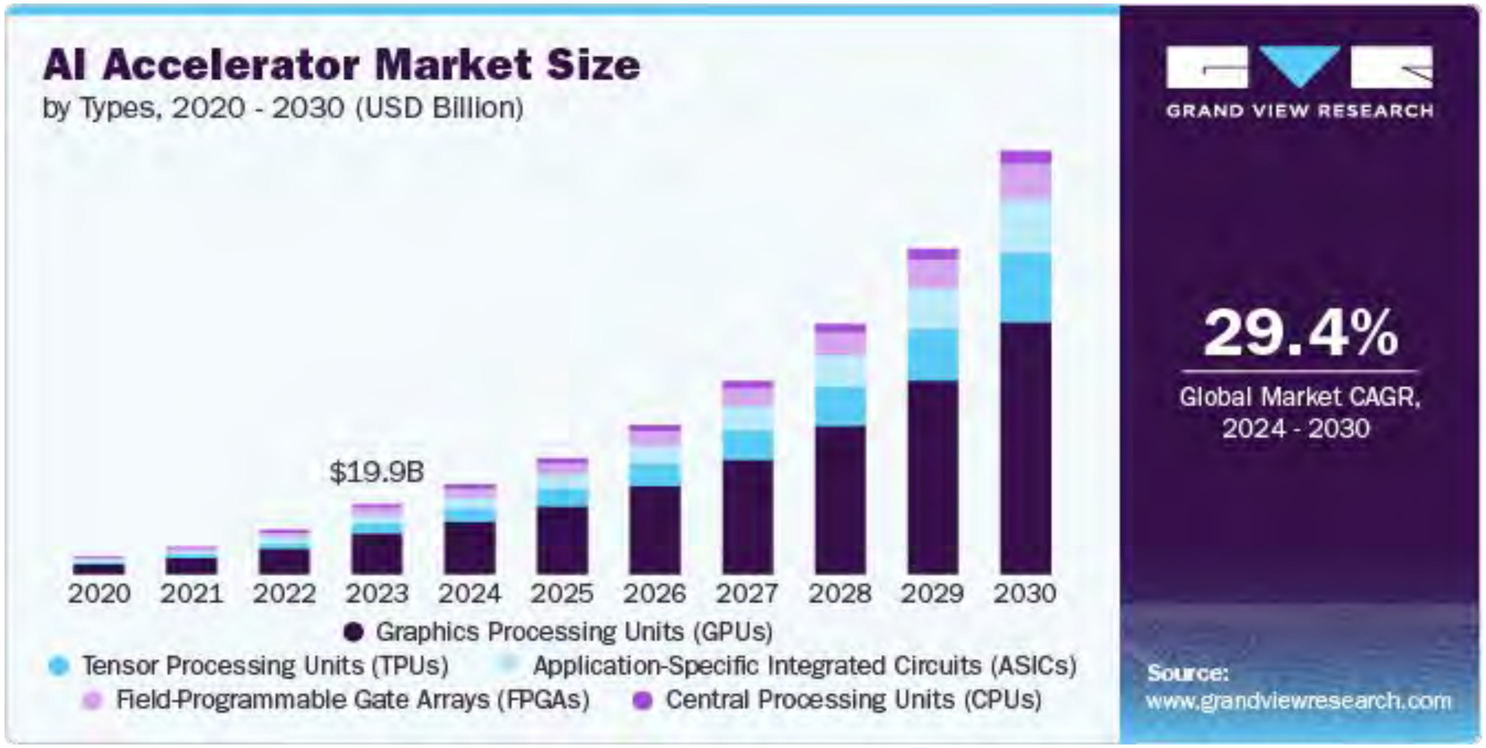
Deep learning models, such as those used in image recognition, natural language processing, and generative AI, demand immense computational power due to their complex architectures and large datasets.

Traditional CPUs, while versatile, often struggle to meet these demands efficiently.

This seminar will dive into the world of hardware acceleration, using specialized processors to speed up deep learning computations and provide a framework for choosing the right hardware for various AI models.

The Compute Boom in AI

- AI training demand doubles every ~5 months, with modern frontier models consuming **4× more FLOPs(Floating Point Operation) each year**, driving exponential growth in compute needs.



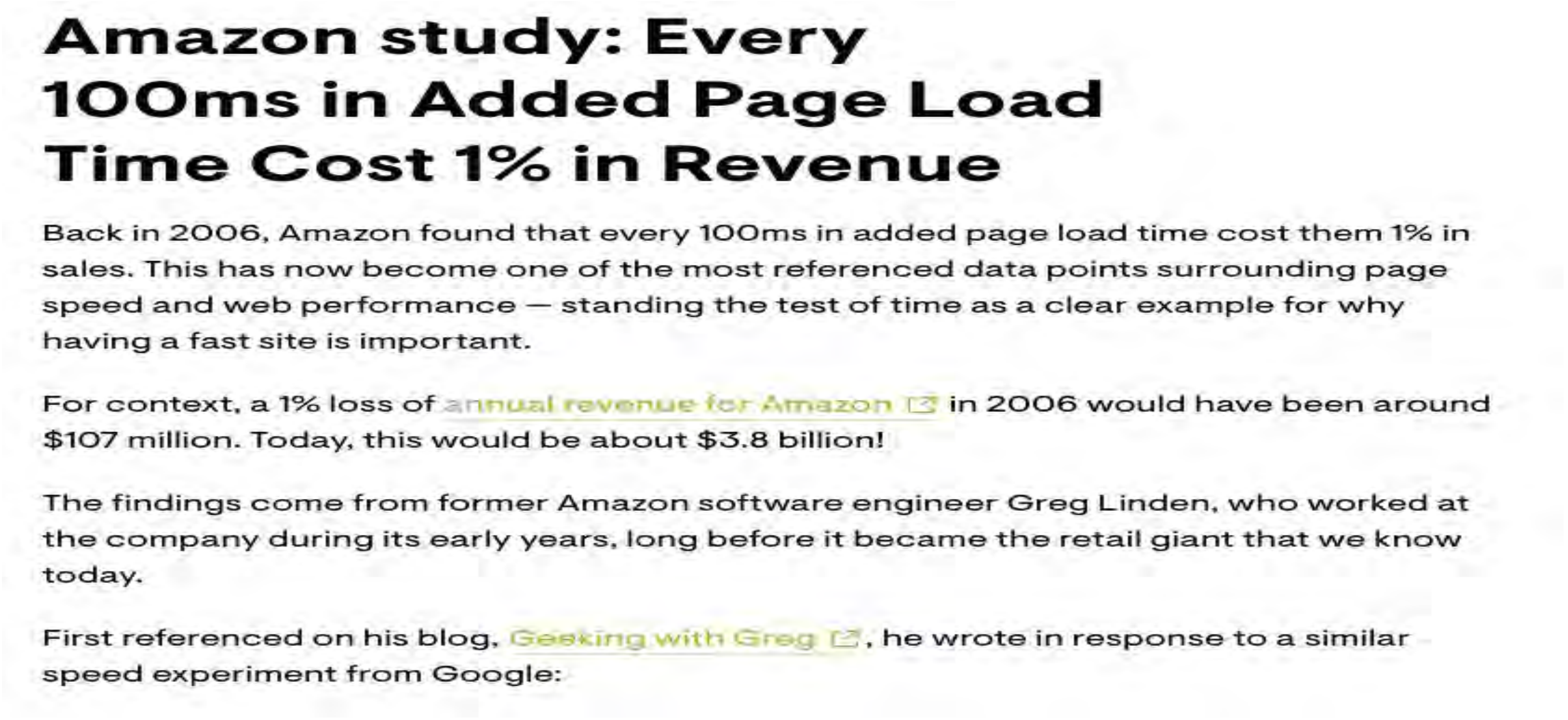
- The AI accelerator market is projected to grow from **$19.9 billion in 2024** to over **$80 billion by 2030**, driven by increasing adoption of GPUs, TPUs, and ASICs, and a **29.4% compound annual growth rate (CAGR)**.

What is an AI Accelerator?

- An AI accelerator is a dedicated processor designed to accelerate machine learning computations.
- Improve performance, reduce latency and reduce cost of deploying machine learning based applications.

Do We Really Need an AI Accelerator for Inference???

- User behaviour studies suggests that people subconsciously notice delays > 100ms and consciously react to anything > 300ms.
- At runtime, prediction dominates response time far more than model loading or data prep.
- A model that takes several hundreds of milliseconds to generate product recommendations, drive users away from your "sluggish", "slow", "frustrating to use" app.



Amazon study: Every 100ms in Added Page Load Time Cost 1% in Revenue

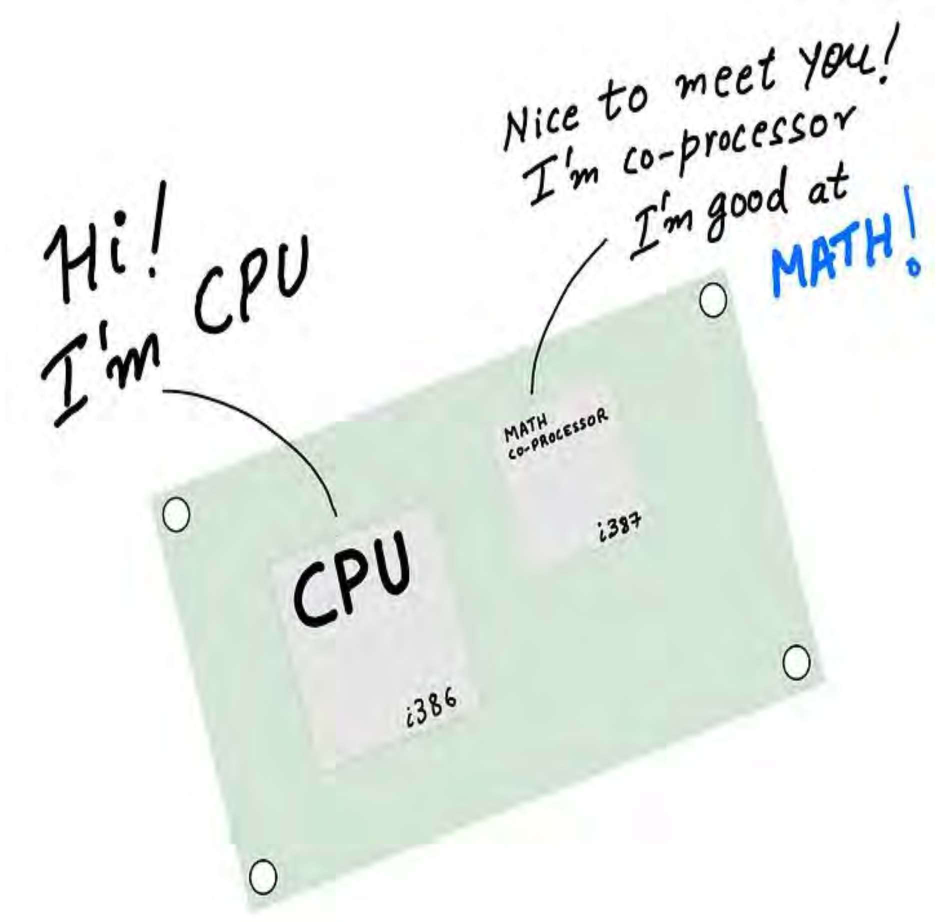
Back in 2006, Amazon found that every 100ms in added page load time cost them 1% in sales. This has now become one of the most referenced data points surrounding page speed and web performance — standing the test of time as a clear example for why having a fast site is important.

For context, a 1% loss of annual revenue for Amazon in 2006 would have been around $107 million. Today, this would be about $3.8 billion!

The findings come from former Amazon software engineer Greg Linden, who worked at the company during its early years, long before it became the retail giant that we know today.

First referenced on his blog, Geeking with Greg, he wrote in response to a similar speed experiment from Google:

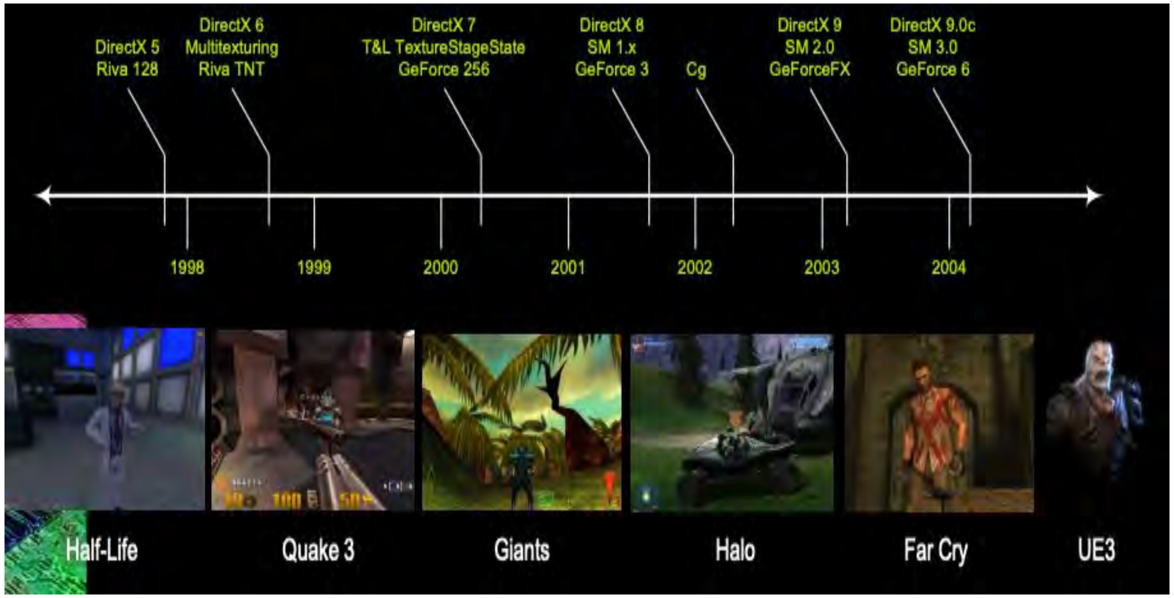
A Brief hardware accelerator history



- 1970s - 80s : CPU paired with FPUs(Math Processors) to offload complex math.
- Known us heterogeneous computing - Multiple Processor types doing different jobs.
- Purpose : Let the CPU run the app, FPU handle the math-heavy lifting.

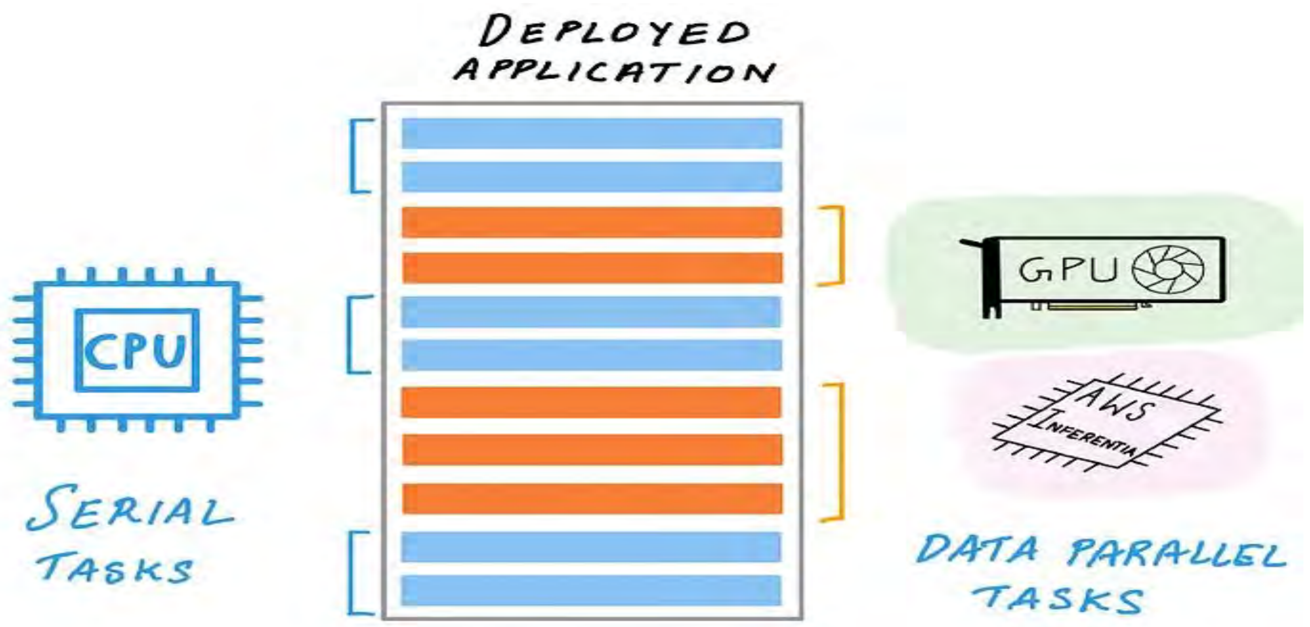
Rise of the GPU & ML needs

- **1990s–2000s**: GPUs emerged to accelerate **graphics** → demand from games & CAD tools.
- **Early 2010s**: Deep learning enters and needs *lots* of matrix math (e.g., CNNs, RNNs).
  - GPUs were already in the market and over the years have become **highly programmable** unlike the early GPUs which were fixed function processors.



Today’s Hardware Acceleration Landscape

- **CPUs** acquired support for **advanced vector extensions (AVX-512)** to accelerate matrix math computations common in deep learning.
- **GPUs** acquired new capabilities such as support for r**educed precision arithmetic (FP16 and INT8)** further accelerating inference.
- **Application Specific Integrated Circuits or ASICs** : purpose-built for deep learning. Eg: **AWS Inferentia**



How to choose the right option?

- “Is GPU better than CPU?”
- “Is CPU cheaper than a GPU?”
- “Is an ASIC always faster than a GPU?”

“Don’t pick hardware based on general rules. Think about *your* model, *your* user expectations, and *your* budget.”

3 Things That Should Guide Your Decision

- **Model Support & Programmability**
  ✔ Large models? ✔ Custom layers? ✔ TensorFlow, PyTorch compatible?

- **Performance vs Cost**
  ✔ Need real-time response? ✔ High throughput? ✔ Budget constraints?

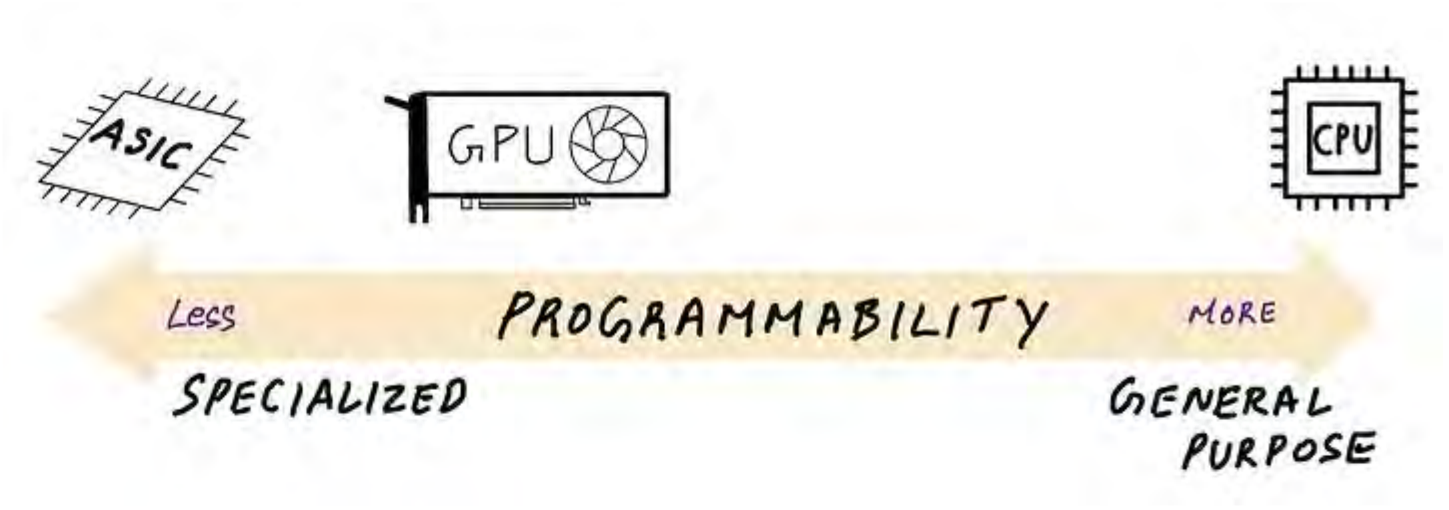
- **Developer Experience / Ease of Use**
  ✔ Easy-to-use toolchain? ✔ Fast learning curve? ✔ Good docs & community?

1.- The Spectrum of Programmability

- **CPUs** are fully programmable and ideal for custom ML code, offering maximum flexibility but lower performance for large models.

- **ASICs**(e.g AWS inferentia) are highly optimized for specific ML tasks with limited programmability, offering top performance at the cost of flexibility.

- **GPUs** are more programmable that ASICs but less so than CPUs, offering strong performance for a wide range of ML workloads



Contd.

- Most AI accelerators can’t automatically accelerate custom code written in high level languages and therefore that piece of code will fall back to CPUs for execution, reducing the overall inference performance.

- NVIDIA GPUs have the advantage that if you want more performance out of your custom code you can reimplement them using CUDA(Compute Unified Device Architecture) programming language and run them on GPUs.

- CUDA is essentially a programming language that extends C and C++ with a few keywords to enable GPU-accelerated computations.

2.- Target throughput, latency and cost

Specialized Processors (e.g., AWS Inferentia)

- Optimized for specific tasks; lower price/performance ratio and better latency than general-purpose CPUs.

GPUs: High Throughput Champions

- Ideal for batch processing where latency is less critical.
- Keeps utilization high, offering cost-effectiveness in cloud environments.

CPUs: Cost-Effective for Real-Time Inference

- Best for smaller models where real-time inference is needed.
- Most cost-effective if latency requirements are met.

3.- Compiler and runtime toolchain and ease of use

CPU Deployments:

- Most deep learning frameworks handle CPU optimizations internally; no extra tools needed.

GPU Optimization with NVIDIA TensorRT:

- Beyond default framework acceleration, using NVIDIA TensorRT can significantly enhance GPU performance through deeper optimization techniques like graph simplification and quantization.

AWS Inferentia via AWS Neuron SDK:

- Tailored optimizations for AWS Inferentia to maximize efficiency.

Real-World AI/App Uses

CPUs: Cost-Effective for Low-Traffic & Flexible Workloads

- Used in LinkedIn's job recommendation alerts or Grammarly's grammar correction, where real-time responses are needed for smaller models and lower traffic.
- Frameworks like PyTorch and TensorFlow optimize for CPUs automatically, no additional tools needed.
- Ideal starting point in the cloud, with easy scalability to GPUs or TPUs when traffic grows.

GPUs: High-Throughput Powerhouses for Scalable AI

- Powering Amazon’s product recommendation systems and Tik Tok video feed ranking, where large batches of data need fast, parallel processing.
- Using NVIDIA TensorRT, companies can optimize models for faster inference through graph simplification and quantization.
- Excellent for cloud environments with burst workloads or scheduled inference jobs.

ASICs (e.g., AWS Inferentia): Specialized for Speed and Efficiency

- Used in real-time fraud detection by fintech apps like Chime or PayPal, where latency and efficiency are critical.
- AWS Neuron SDK compiles models specifically for Inferentia hardware, achieving low-latency and high throughput.
- Great for Amazon Alexa’s voice recognition or large-scale inference tasks in stable production environments.

Accelerator Option 1 : ***GPU-acceleration for inference***

- GPUs are designed for high-volume, parallel computation which makes them ideal for deep learning inferences.

- Originally they were built for rendering pixels; now repurposed for matrix-heavy ML tasks by becoming more programmable(General-Purpose GPUs), supporting CUDA & deep learning libraries.

- GPU-accelerated applications offload the time-consuming routines and functions (also called hotspots) to run on GPUs and take advantage of massive parallelism. The rest of the application still runs on the CPU.

The AlexNet Moment (2012)

- A deep CNN trained on NVIDIA consumer GPUs won ImageNet — 10× faster and cheaper than CPU clusters.

Contd.

- GPUs excel when you need to process **large batches of data or offline inference that doesn't need real time inference**.

Even with **small batches**, modern GPUs can offer **low prediction latency** — especially models like NVIDIA T4, A10G, or H100. **BUT… Utilization Matters**

  - If requests come **sporadically**, your GPU sits idle between jobs.

Cost vs. Utilization Trade-Off

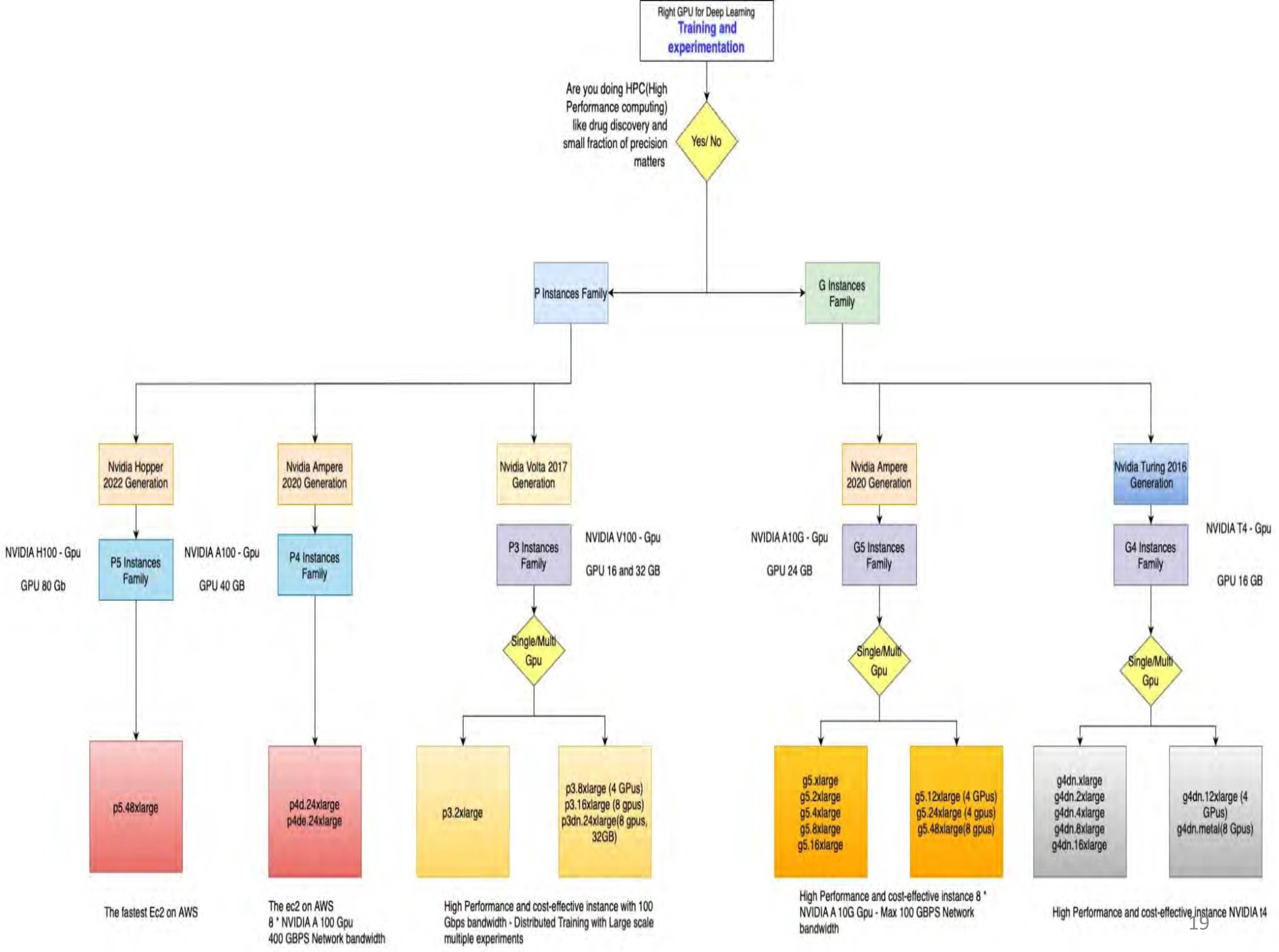
- If GPU is underutilized, **cost per inference goes up**.
- In such cases, **Amazon Elastic Inference** may be more cost-effective — lets you attach lightweight GPU acceleration **on-demand** to a CPU instance.

Choosing the right GPU for deep learning on AWS

- AWS offers **18 EC2 GPU instance types**, each with different GPU models, CPU cores, memory, and bandwidth.

Most Popular Inference GPUs

- **NVIDIA T4 (G4 instances)** – Ideal for *most inference tasks*
  - 16 GB memory
  - Supports FP32, FP16, INT8, Tensor Cores
  - Use **G4 (T4 GPUs)** for standard deep learning inference.
- **NVIDIA V100 (P3 instances)** – Great for *larger models or images*
  - Up to **32 GB memory** on p3dn.24xlarge
  - Upgrade to **P3 (V100 GPUs)** if you need **more memory**, **more throughput**, or are handling **larger data**.

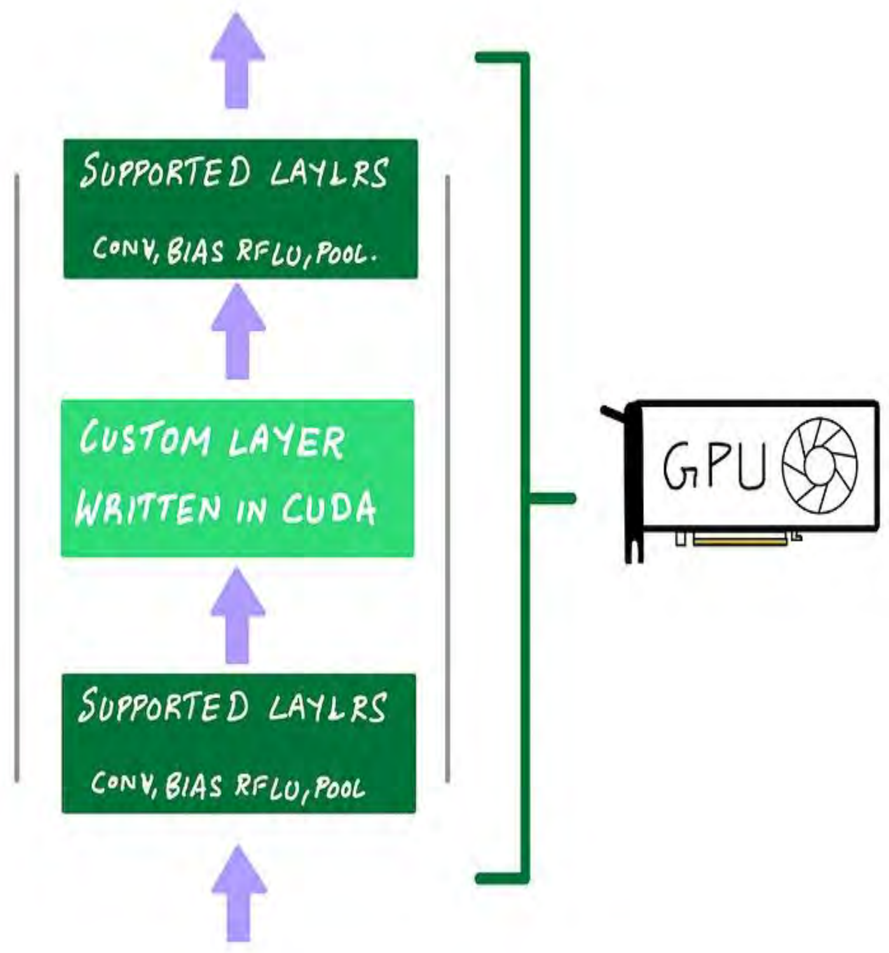

Right GPU for Deep Learning
Training and experimentation
Are you doing HPC(High Performance computing) like drug discovery and small fraction of precision matters
Yes/ No
P Instances Family
G Instances Family
Nvidia Hopper 2022 Generation
Nvidia Ampere 2020 Generation
Nvidia Volta 2017 Generation
Nvidia Ampere 2020 Generation
Nvidia Turing 2016 Generation
NVIDIA H100 - Gpu
GPU 80 Gb
P5 Instances Family
NVIDIA A100 - Gpu
GPU 40 GB
P4 Instances Family
P3 Instances Family
NVIDIA V100 - Gpu
GPU 16 and 32 GB
NVIDIA A10G - Gpu
GPU 24 GB
G5 Instances Family
G4 Instances Family
NVIDIA T4 - Gpu
GPU 16 GB
Single/Multi Gpu
Single/Multi Gpu
Single/Multi Gpu
p5.48xlarge
p4d.24xlarge p4de.24xlarge
p3.2xlarge
p3.8xlarge (4 GPUs) p3.16xlarge (8 gpus) p3dn.24xlarge(8 gpus, 32GB)
g5.xlarge g5.2xlarge g5.4xlarge g5.8xlarge g5.16xlarge
g5.12xlarge (4 GPUs) g5.24xlarge (4 gpus) g5.48xlarge(8 gpus)
g4dn.xlarge g4dn.2xlarge g4dn.4xlarge g4dn.8xlarge g4dn.16xlarge
g4dn.12xlarge (4 GPUs) g4dn.metal(8 Gpus)
The fastest Ec2 on AWS
The ec2 on AWS 8 * NVIDIA A 100 Gpu 400 GBPS Network bandwidth
High Performance and cost-effective instance with 100 Gbps bandwidth - Distributed Training with Large scale multiple experiments
High Performance and cost-effective instance 8 * NVIDIA A 10G Gpu - Max 100 GBPS Network bandwidth
High Performance and cost-effective instance NVIDIA t4

CUDA Programming Model

- **Programmability with performance is one of GPUs greatest strengths.**
- *AlexNet (2012)* was trained with hand-coded CUDA kernels → Sparked the deep learning revolution.

No Low-Level Code Needed:

- Libraries like **cuDNN** and **cuBLAS** do the heavy lifting.
- Frameworks like **PyTorch** & **TensorFlow** use these under the hood — GPU acceleration works automatically.

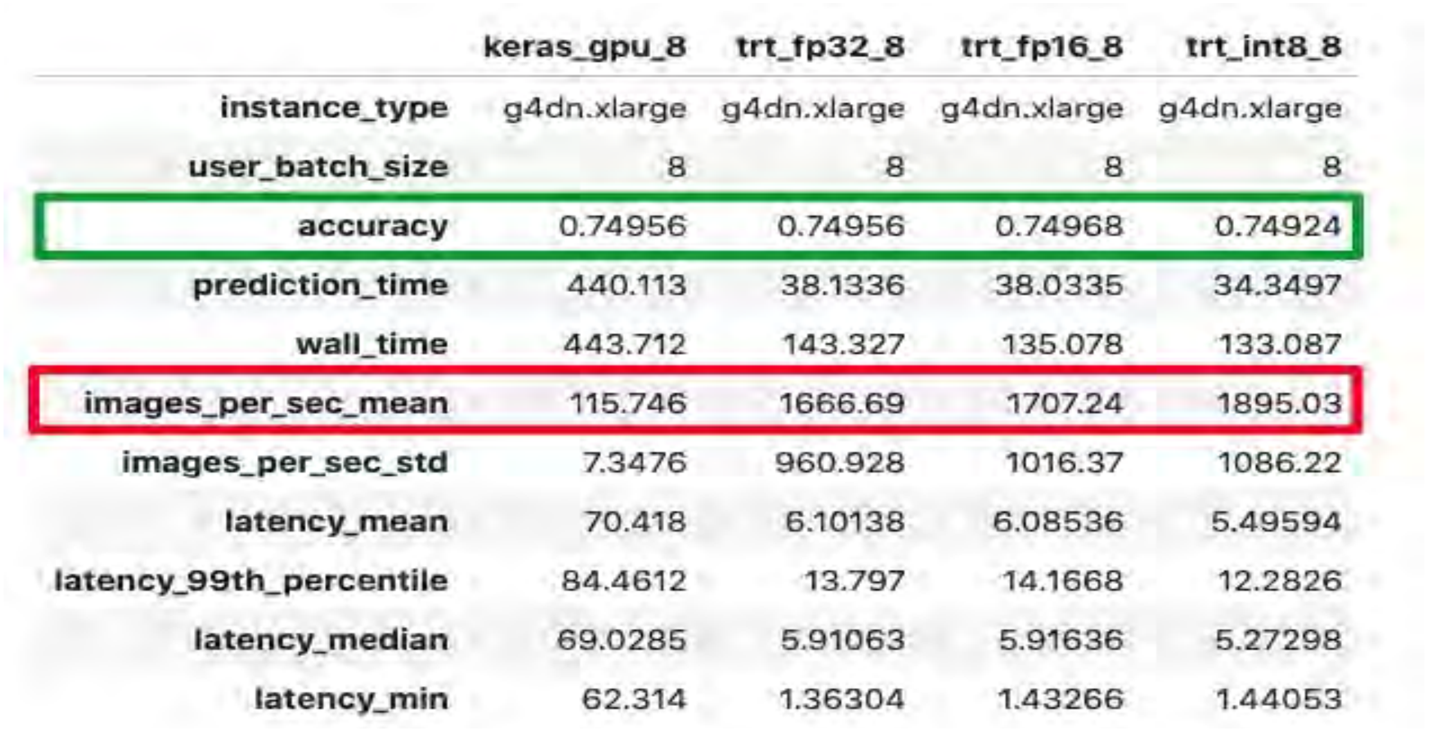


Turning a Good GPU into a Great One

- NVIDIA offers TensorRT, takes a *trained* model and rebuilds it for speed.
- The Key optimization to note are :
  - **Quantization**: reduce model precision from FP32 (single precision) to FP16 (half precision) or INT8 (8-bit integer precision), thereby speeding up inference due to reduced amount of computation
  - **Graph fusion**: merges consecutive layers into one CUDA kernel, slashing kernel-launch overhead.
- Labs and AWS tests regularly see **2–10× higher throughput** vs. "plain" TensorFlow/PyTorch GPU inference, with little or no hit to accuracy.

Case Study: ResNet-50 on g4dn.xlarge

- Negligible accuracy drop (< 0.5%) even at INT8
- Throughput soars ~ 10x from 115 to 1000 +imgs/s => far lower cost per request.



	keras_gpu_8	trt_fp32_8	trt_fp16_8	trt_int8_8
instance_type	g4dn.xlarge	g4dn.xlarge	g4dn.xlarge	g4dn.xlarge
user_batch_size	8	8	8	8
accuracy	0.74956	0.74956	0.74968	0.74924
prediction_time	440.113	38.1336	38.0335	34.3497
wall_time	443.712	143.327	135.078	133.087
images_per_sec_mean	115.746	1666.69	1707.24	1895.03
images_per_sec_std	7.3476	960.928	1016.37	1086.22
latency_mean	70.418	6.10138	6.08536	5.49594
latency_99th_percentile	84.4612	13.797	14.1668	12.2826
latency_median	69.0285	5.91063	5.91636	5.27298
latency_min	62.314	1.36304	1.43266	1.44053

https://github.com/shashankprasanna/ai-accelerators-examples/blob/main/gpu-tf-tensorrt-resnet50.ipynb

Why it matters?

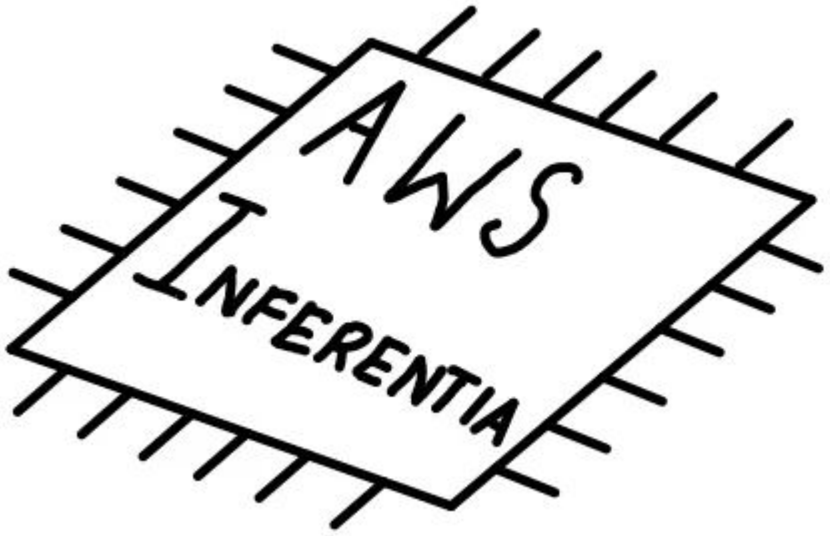
TensorRT can increase each GPU's throughput by up to ~10× (in the ResNet-50 example, from ~115 to > 1000 images/second). If every GPU now processes ten times more requests, you need only **one-tenth as many GPUs** to handle the same overall traffic load.

- **Lower cloud bill** – fewer GPU hours to pay for.
- **Smaller carbon footprint & power draw**.
- **Simpler ops** – less hardware to deploy, monitor, and scale.

Accelerator Option 2 : ***AWS Inferentia for inference***

- AWS Inferentia is an **ASIC** (Application-Specific Integrated Circuit) custom silicon built by Amazon to run deep learning **inference workloads**.

- Like the chip in your **noise-canceling headphones** or a DVD decoder, it's specialized: **1 job, done very efficiently**.



Contd.

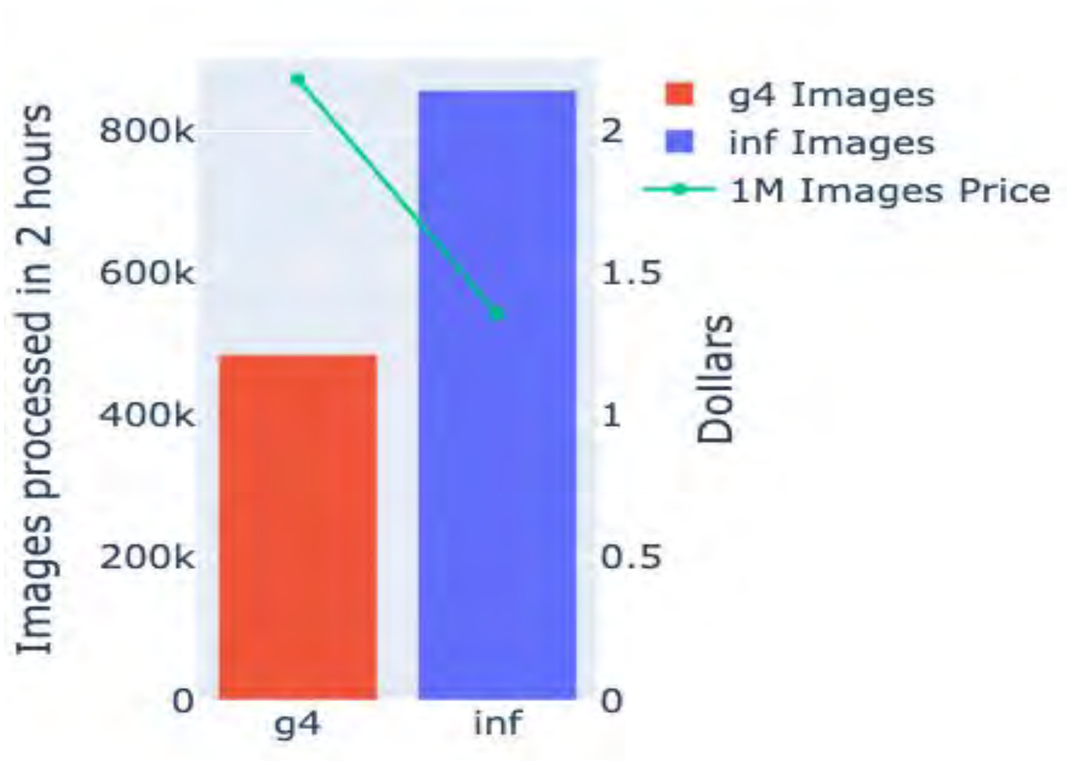
- AWS Inferentia was purpose built to offer high inference performance at the lowest cost in the cloud.
- Available via **EC2 Inf1 instances**: from 1 to 16 Inferentia chips per instance.

In most cases, AWS Inferentia might be the best AI accelerator for your use case, if your model:

- Was trained in MXNet, TensorFlow, PyTorch or has been converted to ONNX
- Has operators that are supported by the AWS Neuron SDK.

AWS Inferentia throughput, latency and cost

- AWS has compared performance of AWS Inferentia vs. GPU instances for popular models, and reports lower cost for popular models: [YOLOv4 model]().



- g4 processes **≈ 480 k images** in two hours.
- inf1 processes **≈ 860 k images** in the same time.
  → **1.85× higher throughput** for Inferentia.
- The green line converts the on-demand hourly price of each instance into “dollars to process 1 M images.”
- g4 ≈ $2.40 per million images
- inf1 ≈ $1.50 per million images
  → **~37 % lower cost** with Inferentia.

AWS inferentia supported models, operators and precisions

- AWS Inferentia only supports a **specific set of neural network operators** exposed through the **Neuron SDK**.

When compiling, the Neuron SDK:

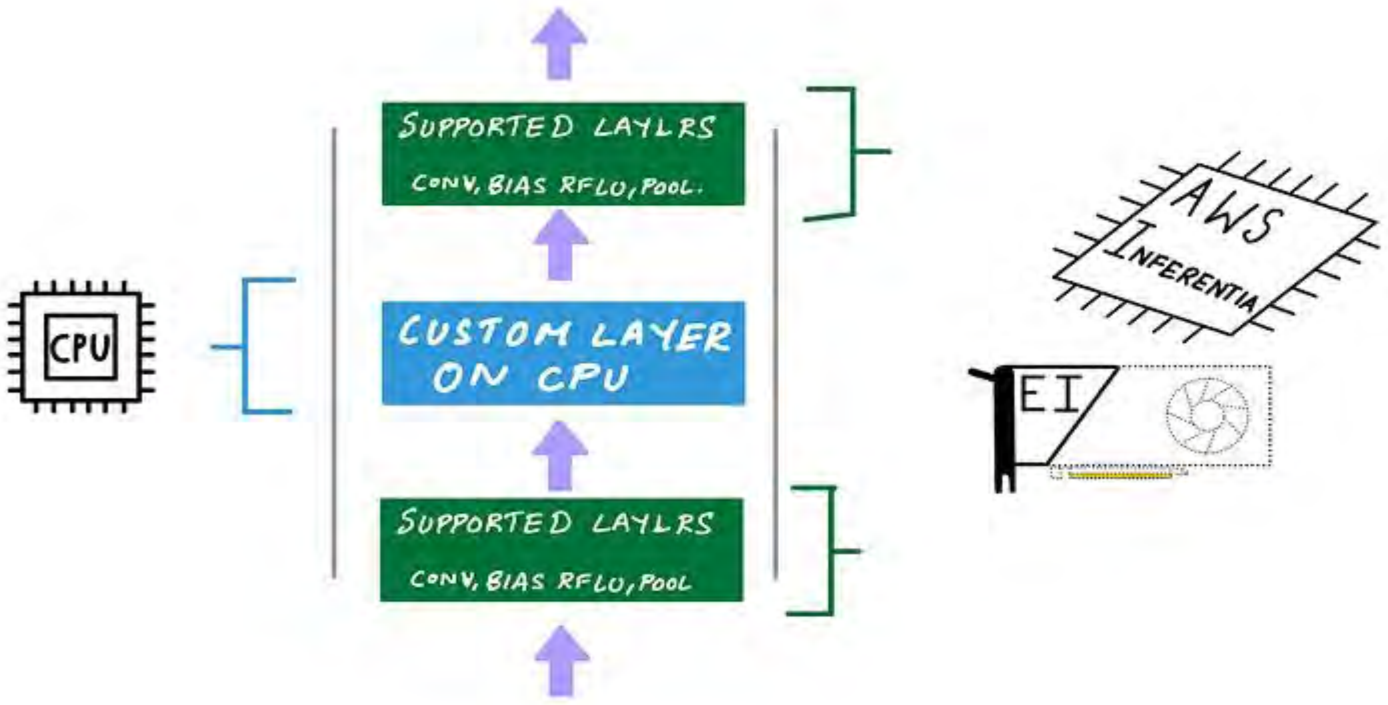
- **Checks operator compatibility**
- **Splits** the model into **Inferentia-accelerated parts** and **CPU-handled parts** if needed.

Precision Handling

- **FP32 models** are **automatically cast to BF16** for faster performance.
- **FP16 models** are used directly without conversion.
- **INT8** is **hardware-supported**, but **not currently deployable** via Neuron SDK.

What Happens If…

- An op is **not supported ➜** it runs on the **host CPU**.
- A supported op is **more efficient on CPU ➜** it's **routed to CPU** automatically.



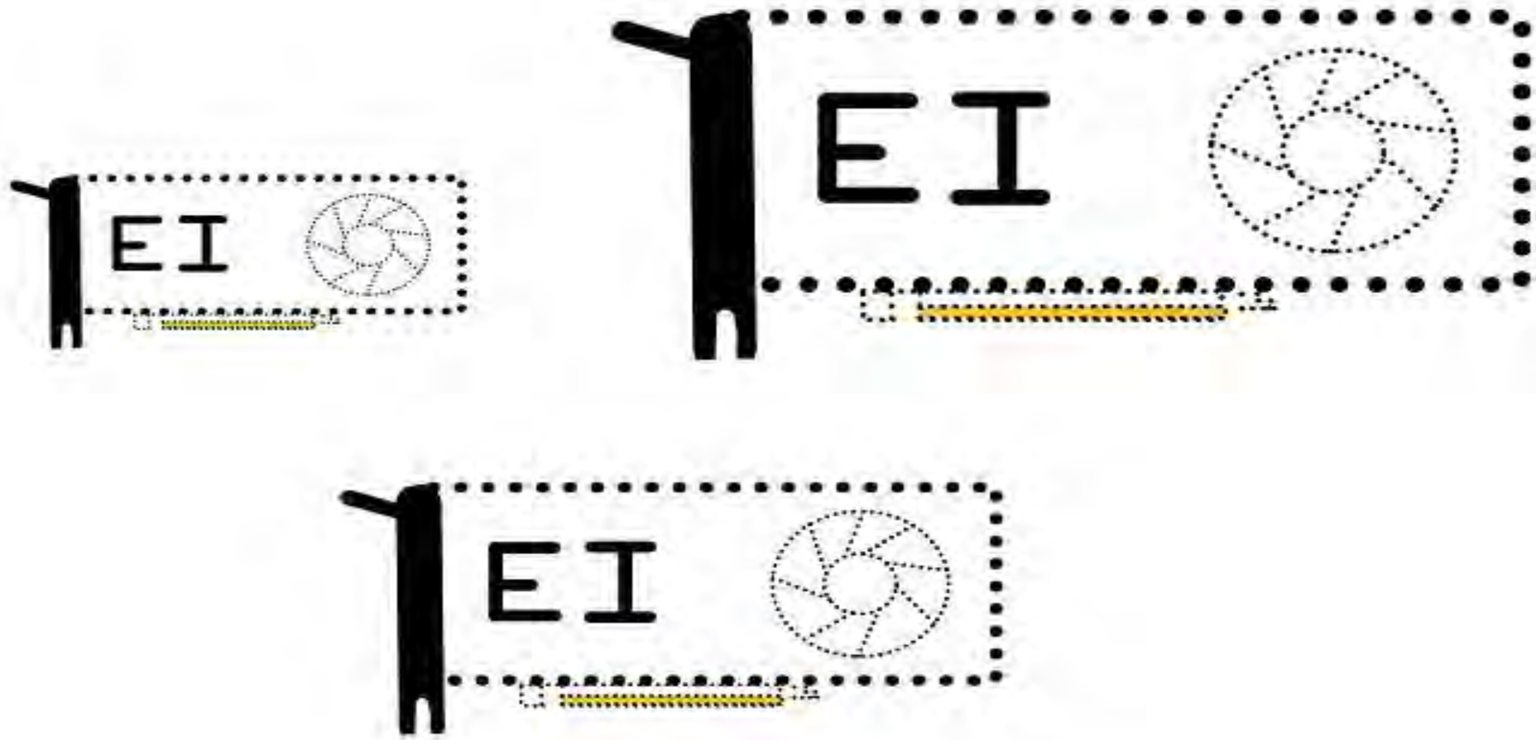
AWS Inferentia - Flexibility & Fine-Grained Control

- Inferentia offers more control per core than a GPU — letting you fine-tune how the model runs, layer by layer, across dedicated engines.

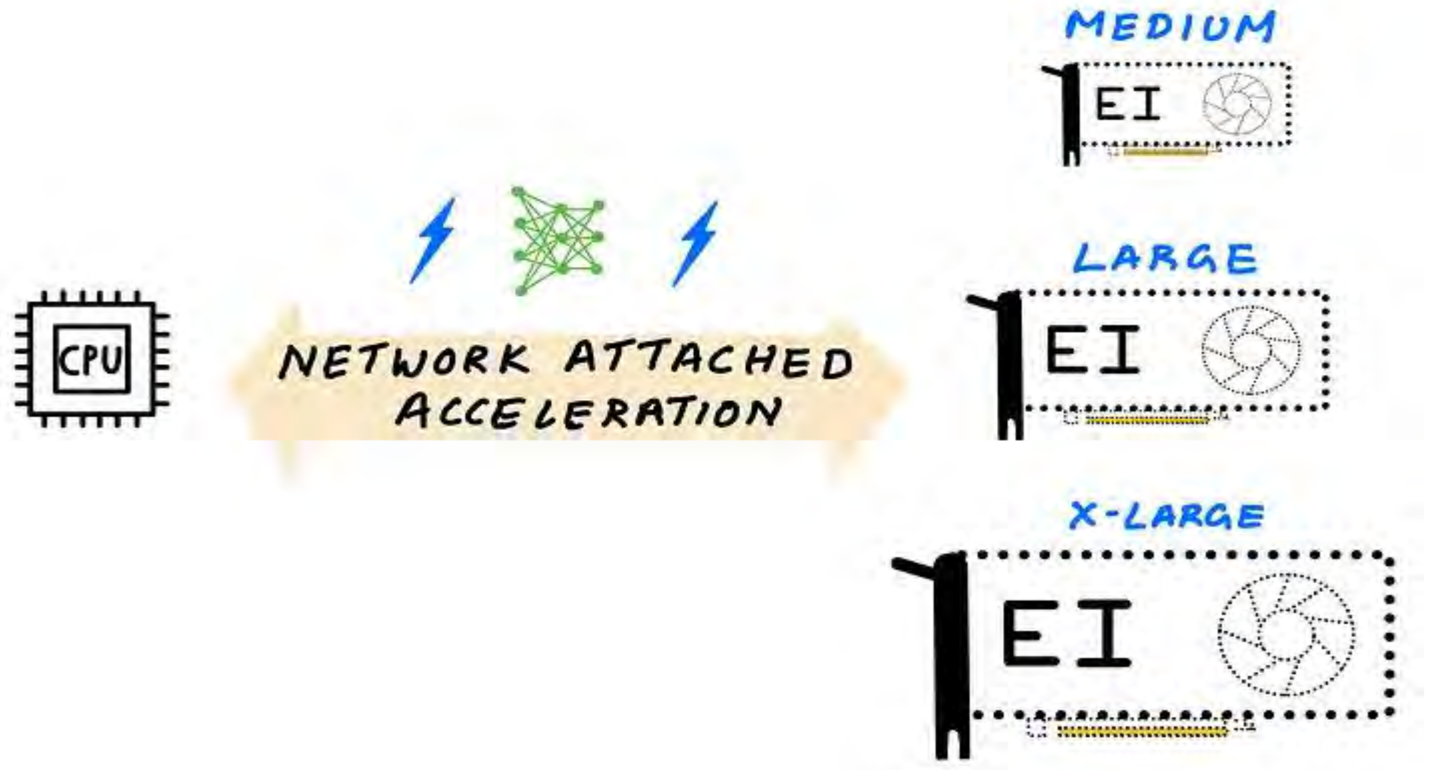
- Using techniques like:
  - **Batching** – Maximize Throughput
    - During compilation, you can specify a **batch size** to enable **weight caching** and reduce memory overhead.

  - **Pipelining –** Reduce Latency
    - Use multiple NeuronCores to **cache different model layers in parallel**.

Accelerator Option 3 : *Amazon Elastic Inference (EI) acceleration for inference*



- Amazon Elastic Inference (EI) allows you to add cost-effective variable-size GPU acceleration to a CPU-only instance without provisioning a dedicated GPU instance.
- No need to provision a full GPU instance which saves cost.
- EI accelerator is attached through the network using an **AWS PrivateLink endpoint service**



Why Choose Amazon EI Over Dedicated GPU Instances?

Smart, Scalable Inference for Small Models

- GPUs are **throughput-oriented** — great for big batches, but often **underutilized** in real-time, small-batch scenarios.
- With **EI**, you attach just enough GPU acceleration to a CPU instance — no need to over-provision a full GPU.

Performance vs Cost Trade-off

- EI adds **some latency** due to network attachment
- But it's **faster than CPU-only**, **cheaper than GPUs**, and often **"good enough"** to meet SLA (e.g., <200 ms latency).

Final Takeaway- Choosing the Right Inference Accelerator

No One-Size-Fits-All – Choose Based on Your Needs:

- **Model Type & Programmability**
  - Need custom ops? → GPU with CUDA
  - Simpler model, low performance need? → CPU might be enough
  - Supported model, no custom logic? → Inferentia or Elastic Inference saves cost
- **Target Throughput, Latency, and Cost**
  - Real-time, high-volume? → GPU or Inferentia
  - Cost-sensitive, moderate latency? → Elastic Inference
- **Ease of Use**
  - Native frameworks = simplest
  - Slight learning curve with **Neuron SDK** (Inferentia) or **TensorRT** (GPU) can lead to **big cost savings**

Reference

- The Evolution of GPUs for General Purpose Computing
- NVIDIA Data Center Deep Learning Product Performance
- Achieving 1.85x higher performance for deep learning based object detection with an AWS Neuron compiled YOLOv4 model on AWS Inferentia
- Efficient Processing of Deep Neural Networks: A Tutorial and Survey

Thank you!!

SOUTH PARK
BACKCOUNTRY ADVENTURES
ANYBODY HAVE ANY QUESTIONS?