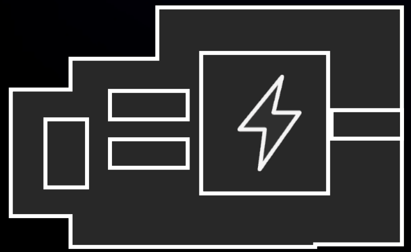
CHIPS & COMPILERS SYMPOSIUM, MLSYS'22

Optimizing ML workloads with AWS Inferentia & Trainium

Tobias Edler von Koch
Sr. Compiler Engineer
AWS

Ron Diamant
Sr. Principal ML Engineer
AWS

Silicon innovation at AWS

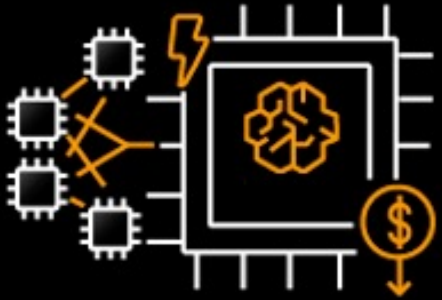


AWS Nitro System

Hypervisor, network,
storage, SSD, and security

AWS Graviton

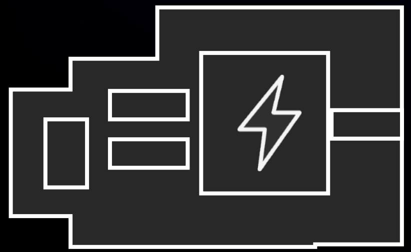
Powerful and efficient,
modern applications



AWS Inferentia and AWS Trainium

Machine learning acceleration

Silicon innovation at AWS

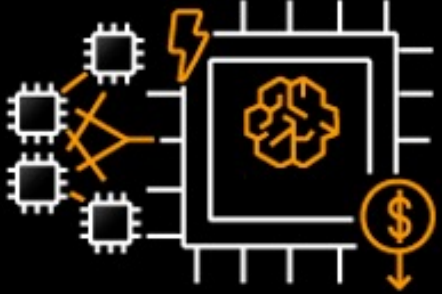


AWS Nitro System

Hypervisor, network, storage, SSD, and security

AWS Graviton

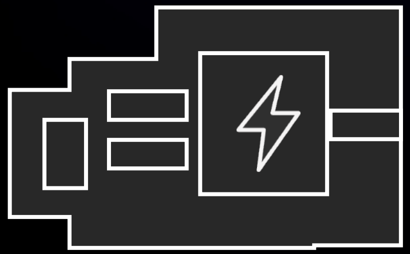
Powerful and efficient, modern applications



AWS Inferentia and AWS Trainium

Machine learning acceleration

Silicon innovation at AWS

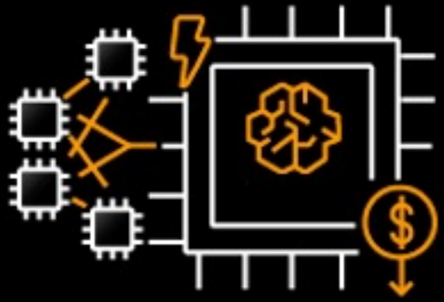


AWS Nitro System

Hypervisor, network,
storage, SSD, and security

AWS Graviton

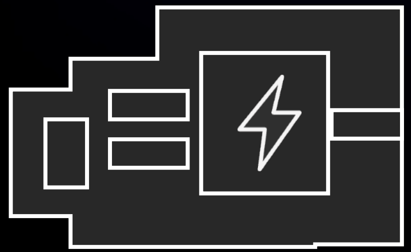
Powerful and efficient,
modern applications



AWS Inferentia and AWS Trainium

Machine learning acceleration

Silicon innovation at AWS

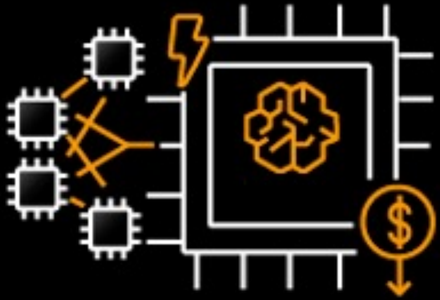


AWS Nitro System

Hypervisor, network,
storage, SSD, and security

AWS Graviton

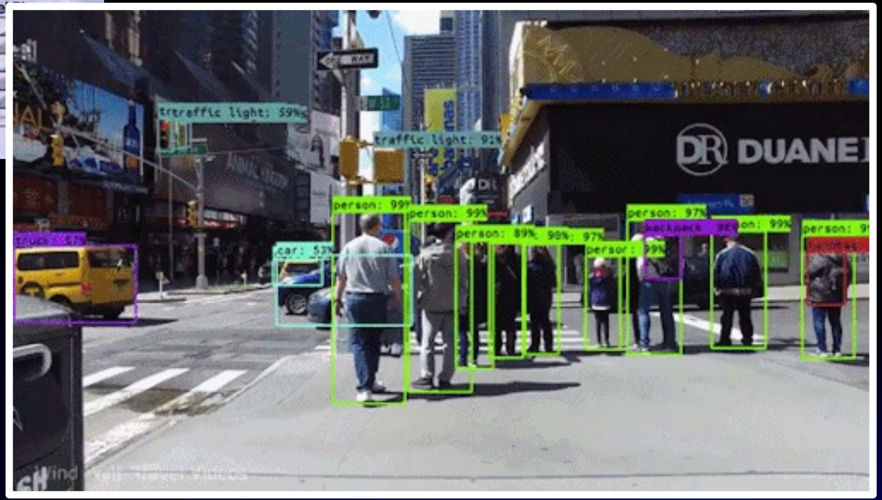
Powerful and efficient,
modern applications



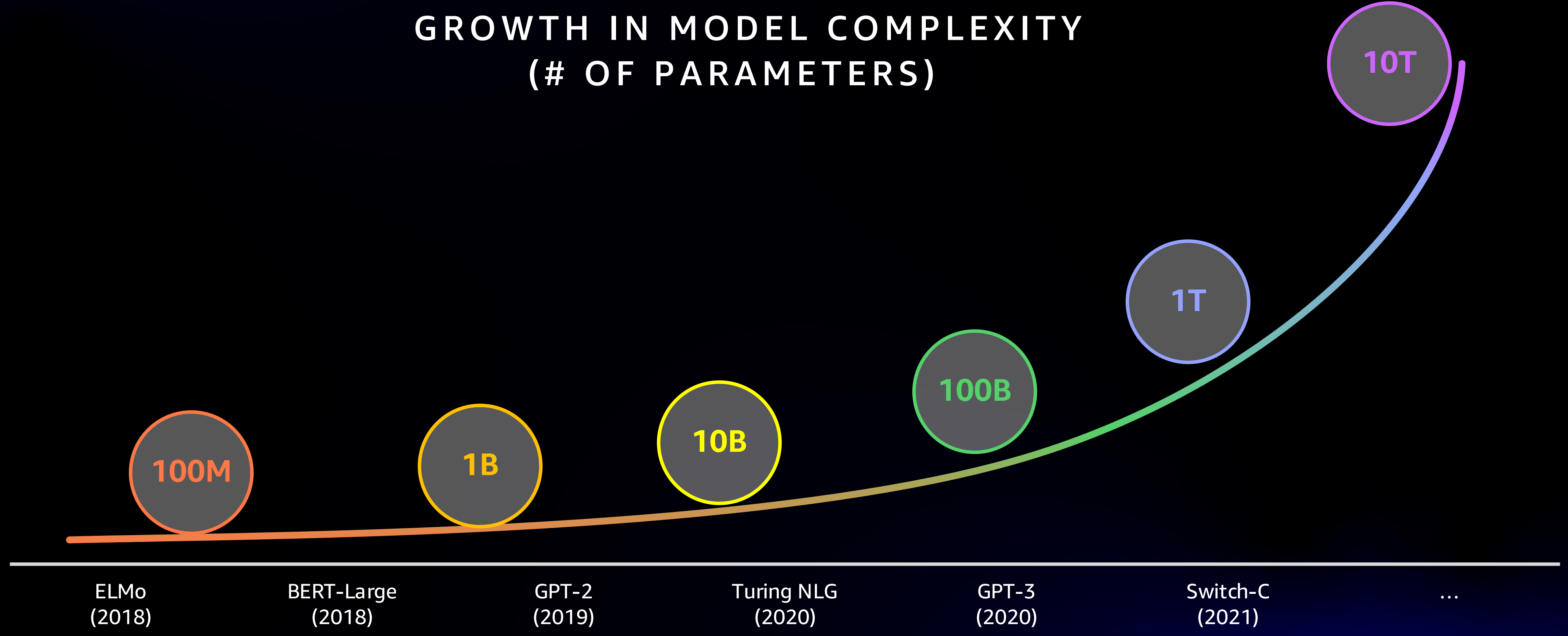
AWS Inferentia and AWS Trainium

Machine learning acceleration

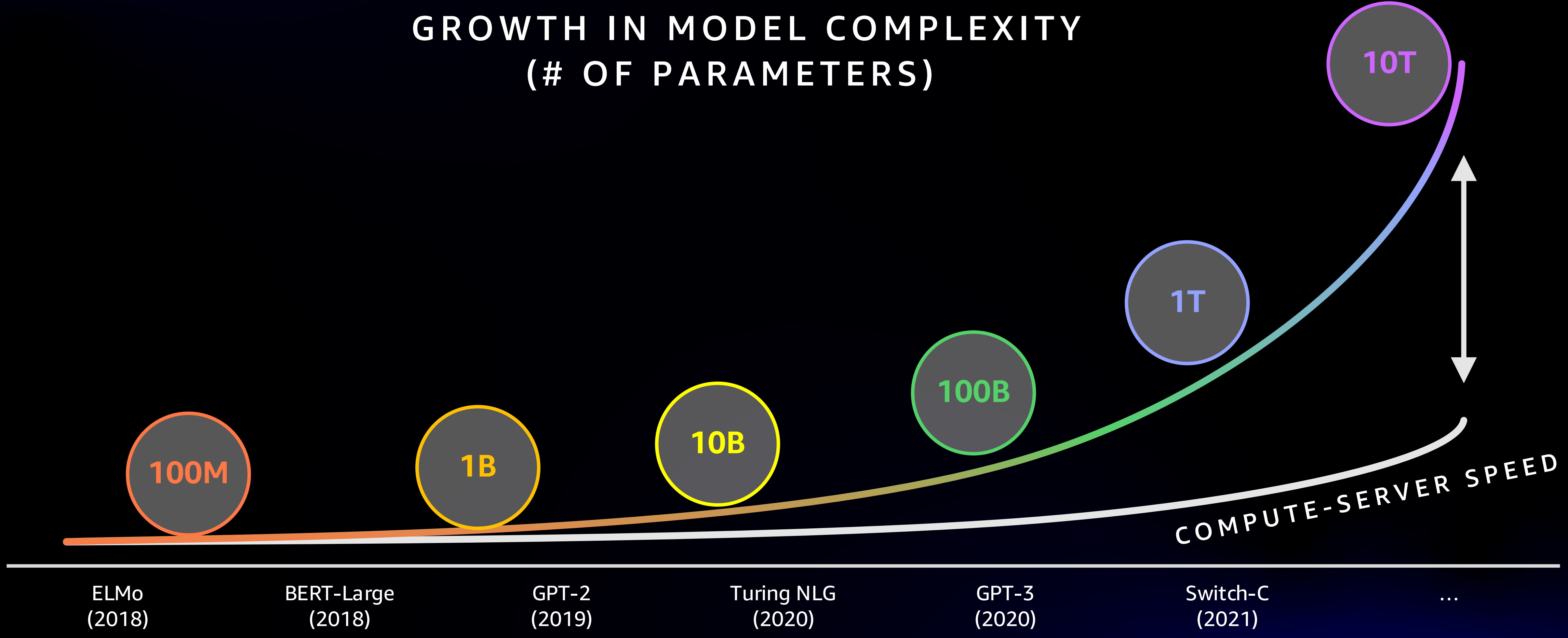
Machine Learning Acceleration



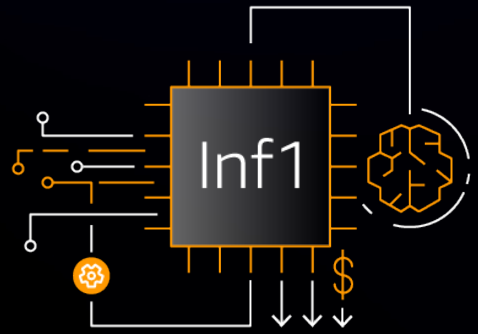
Machine Learning trends



Machine Learning trends

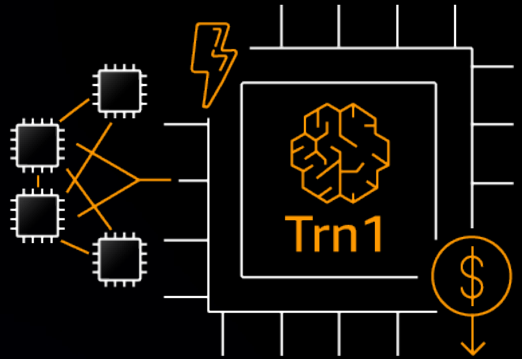


AWS ML Accelerators for Deep Learning



AWS Inf1

Powered by AWS Inferentia

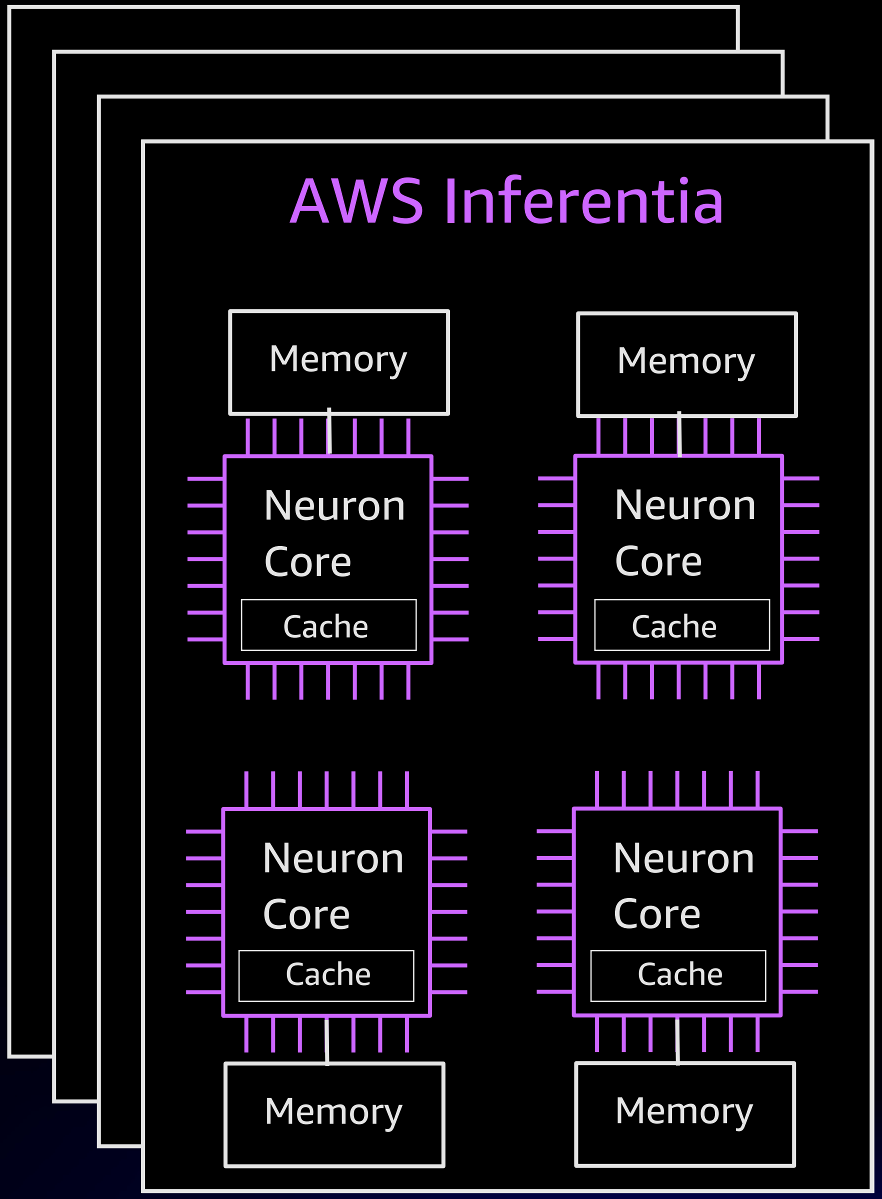


AWS Trn1

Powered by AWS Trainium

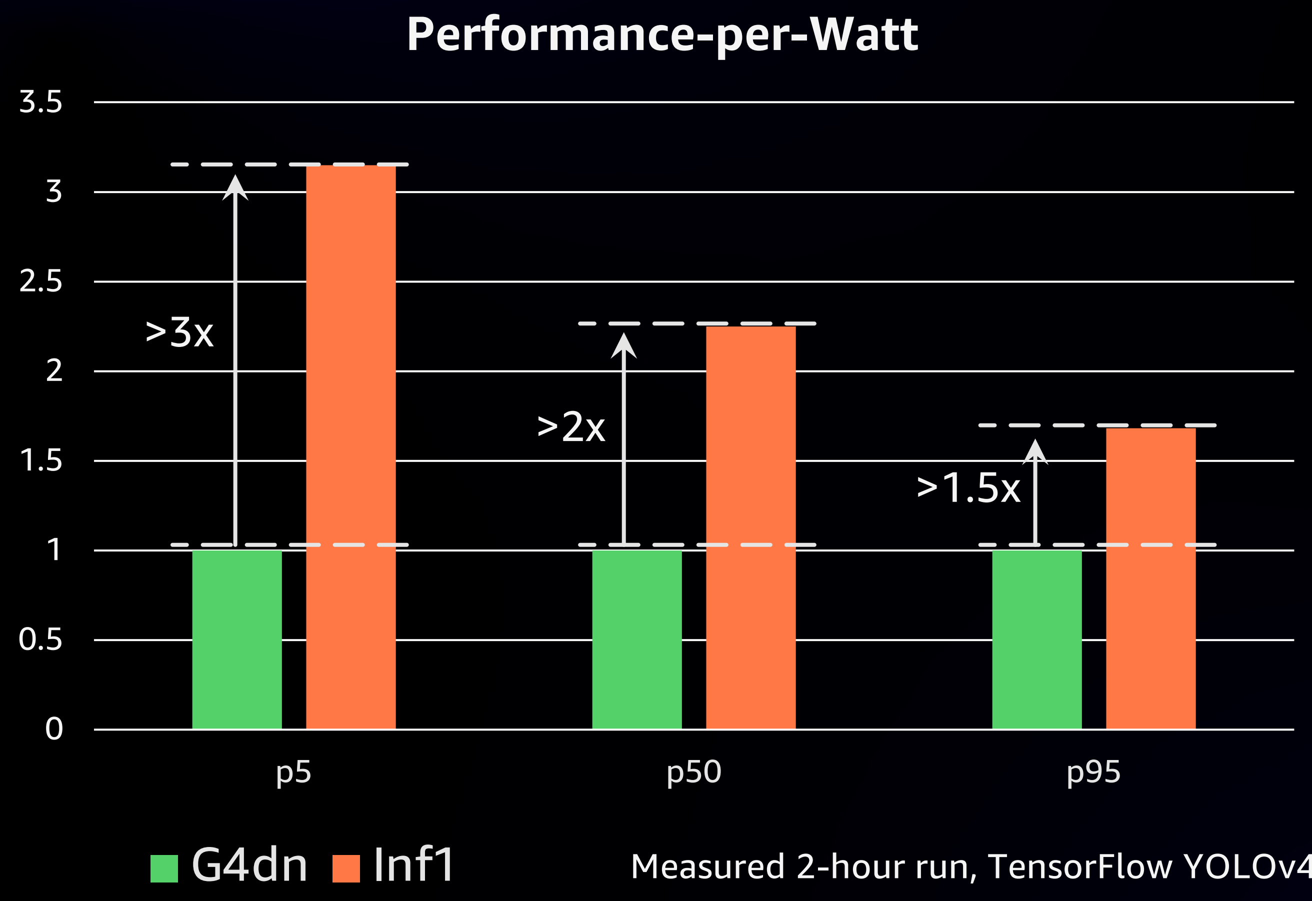
AWS Inferentia

- 4 Neuron Cores
- Up to 128 TOPs/chip
- Co-optimize throughput and latency
  - Large on-chip caches
  - Fast chip-to-chip interconnect
- Ease of use!
  - Supported in popular ML frameworks
  - FP16, BF16, INT8

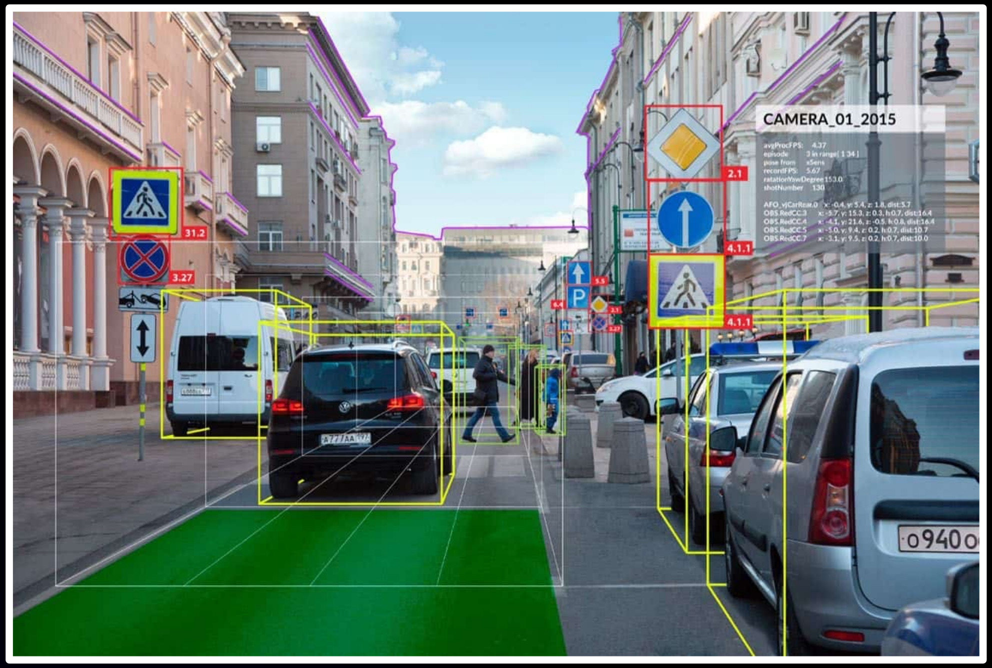


aws

AWS Inferentia – Sustainable performance

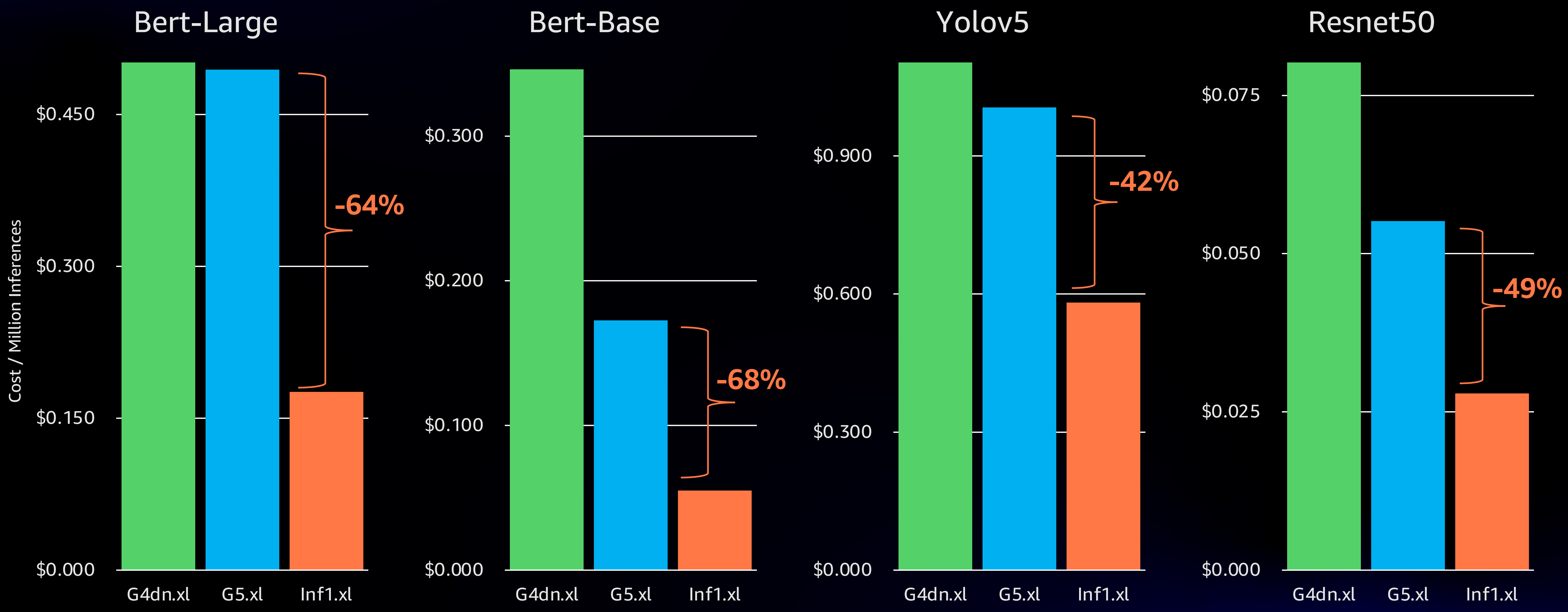


Measured 2-hour run, TensorFlow YOLOv4

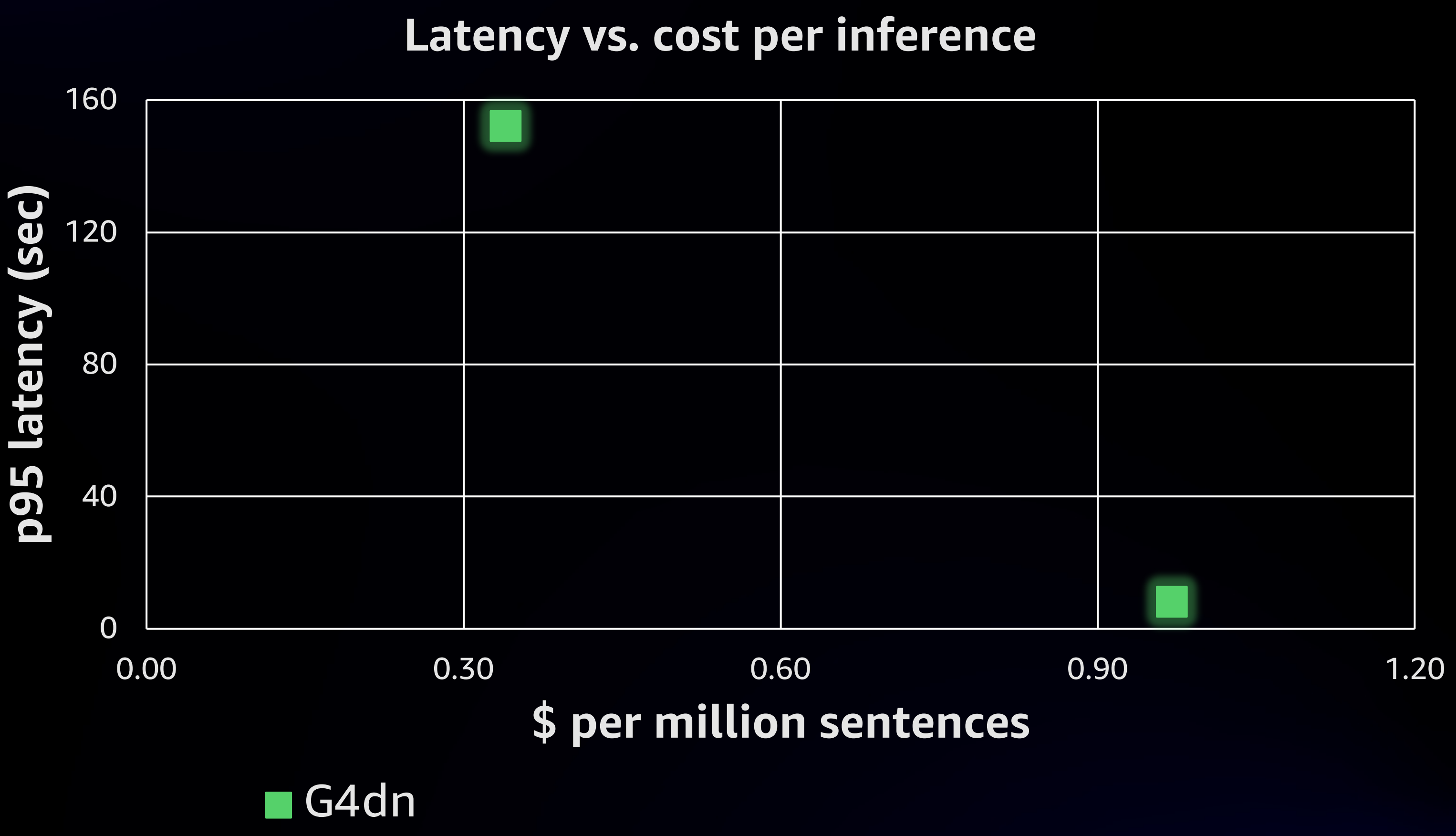


Objects in an image, as detected by YOLOv4

AWS Inferentia – Up to 68% lower cost



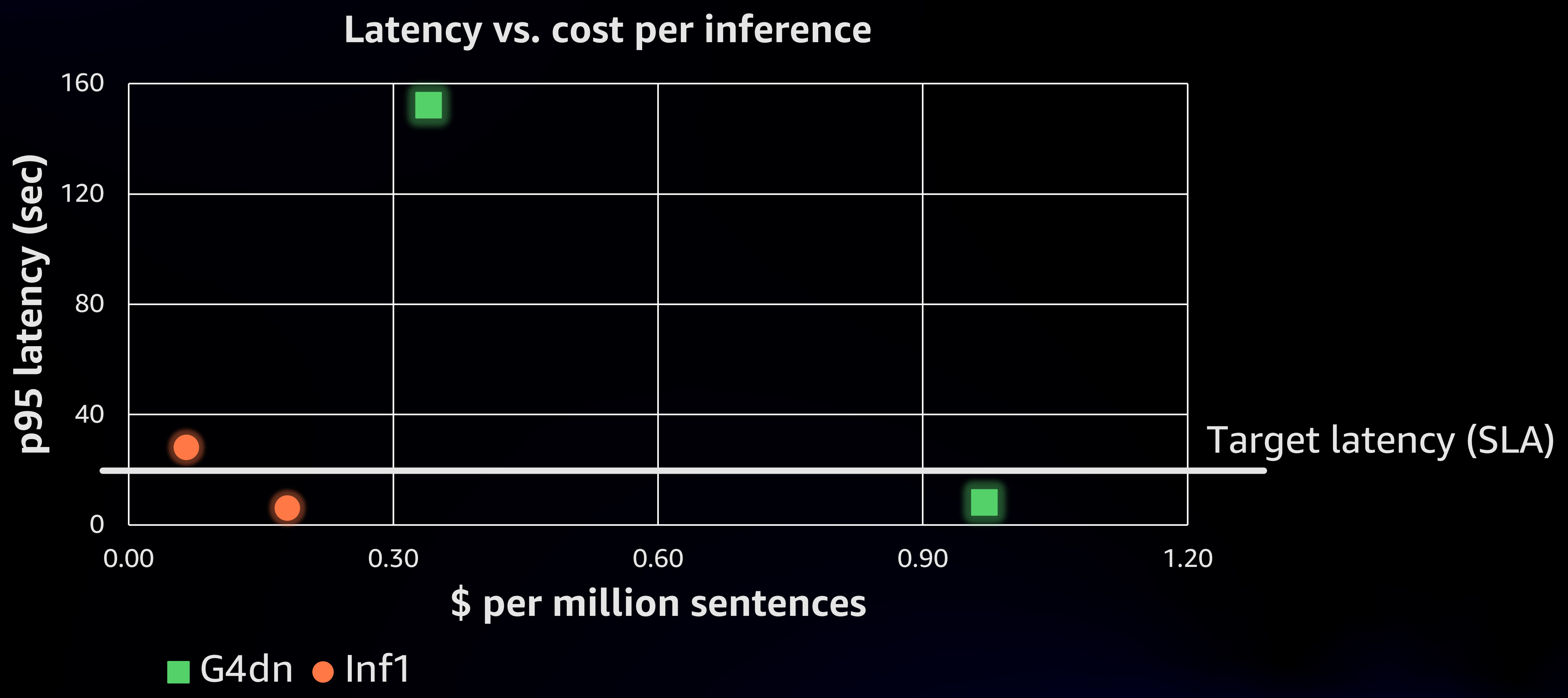
Co-optimizing latency and throughput



Co-optimizing latency and throughput

Co-optimizing latency and throughput

NEURONCORE PIPELINE FOR LATENCY-BOUND APPS



AMAZON ALEXA

Alexa has deployed highly complex text-to-speech model that generates **human-like speech**, to support over **100 million Alexa devices globally**

With Inf1 instances, they have been able **to lower their operating costs by about 30%** over GPU instances, while achieving **25% better inference latency**

AWS Inferentia - Customer adoption

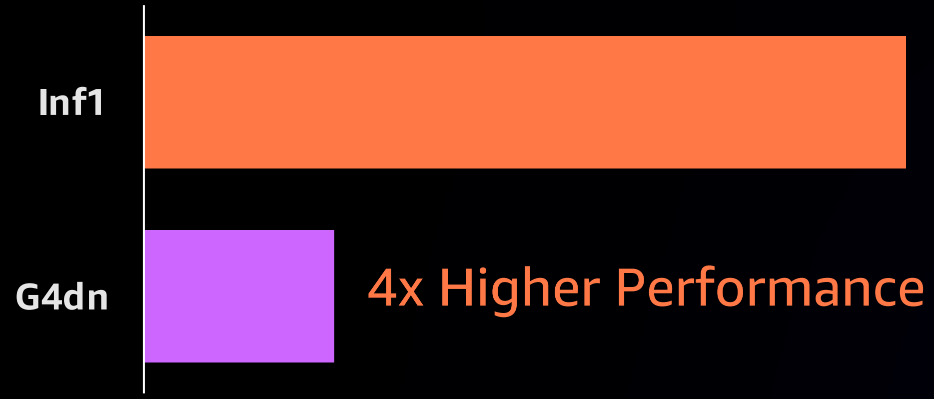
airbnb
AUTODESK
Finch COMPUTING
sprinklr
Amazon Rekognition
amazon robotics
amazon advertising
amazon alexa
prime video
The Asahi Shimbun
CONDÉ NAST
Talroo™
Anthem.
DISCO
SKYWATCH
Forward
NTT PC COMMUNICATIONS
DMP DIGITAL MEDIA PROFESSIONALS
SCREENING EAGLE
HiTTO

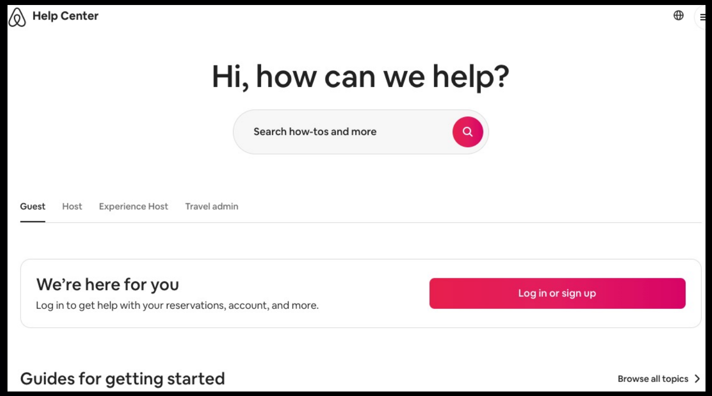
aws

AWS Inferentia - Customer adoption

PyTorch BERT Throughput (Chatbot Engine)

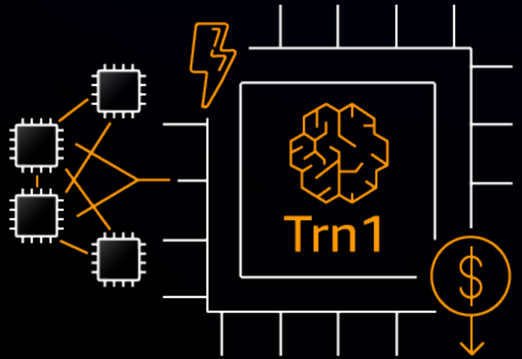
Video Analysis Performance





airbnb

AWS Trainium



AWS Trn1/Trn1n

Powered by AWS Trainium

Trn1

MATH ENGINE FREQUENCY
3 GHz

BF16/FP16	TF32	FP32
3.4 PFLOPS	3.4 PFLOPS	840 TFLOPS

AGGREGATE ACCELERATOR MEMORY
512 GB

PEAK MEMORY BANDWIDTH
13.1 TB/sec

NEURONLINK BANDWIDTH BETWEEN CHIPS
768 GB/sec

NETWORK CONNECTIVITY
800 Gbps EFA
1600 Gpbs EFA

AWS Trainium

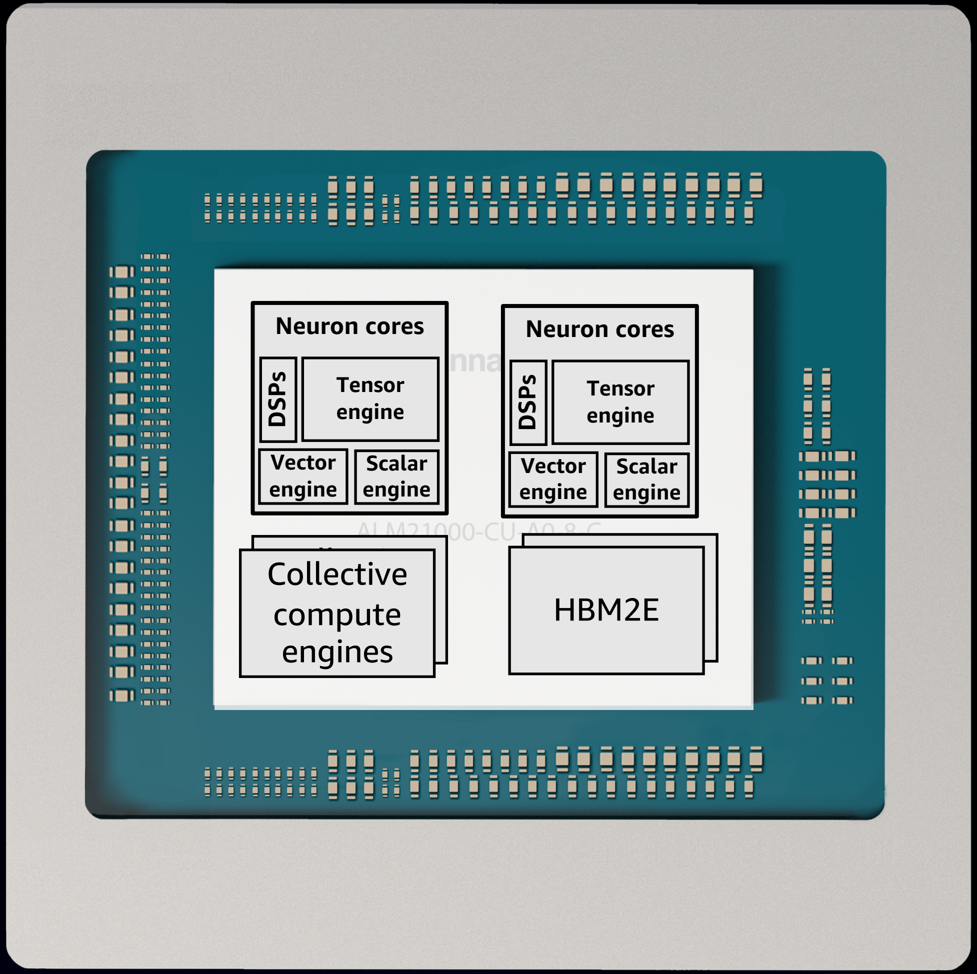
2 NeuronCores

Tensor, scalar, and vector engines

Dedicated collective compute engines

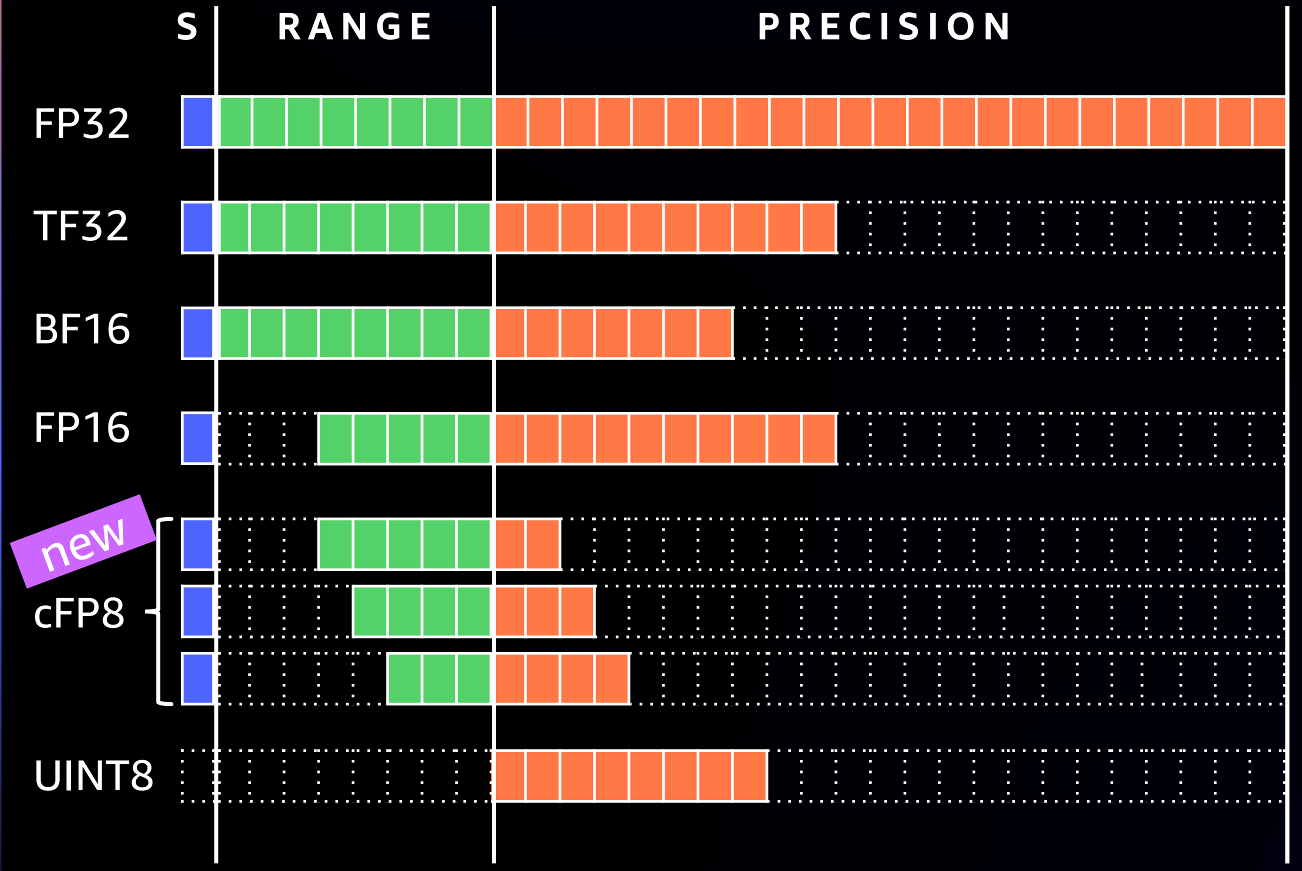
Embedded general purpose DSPs

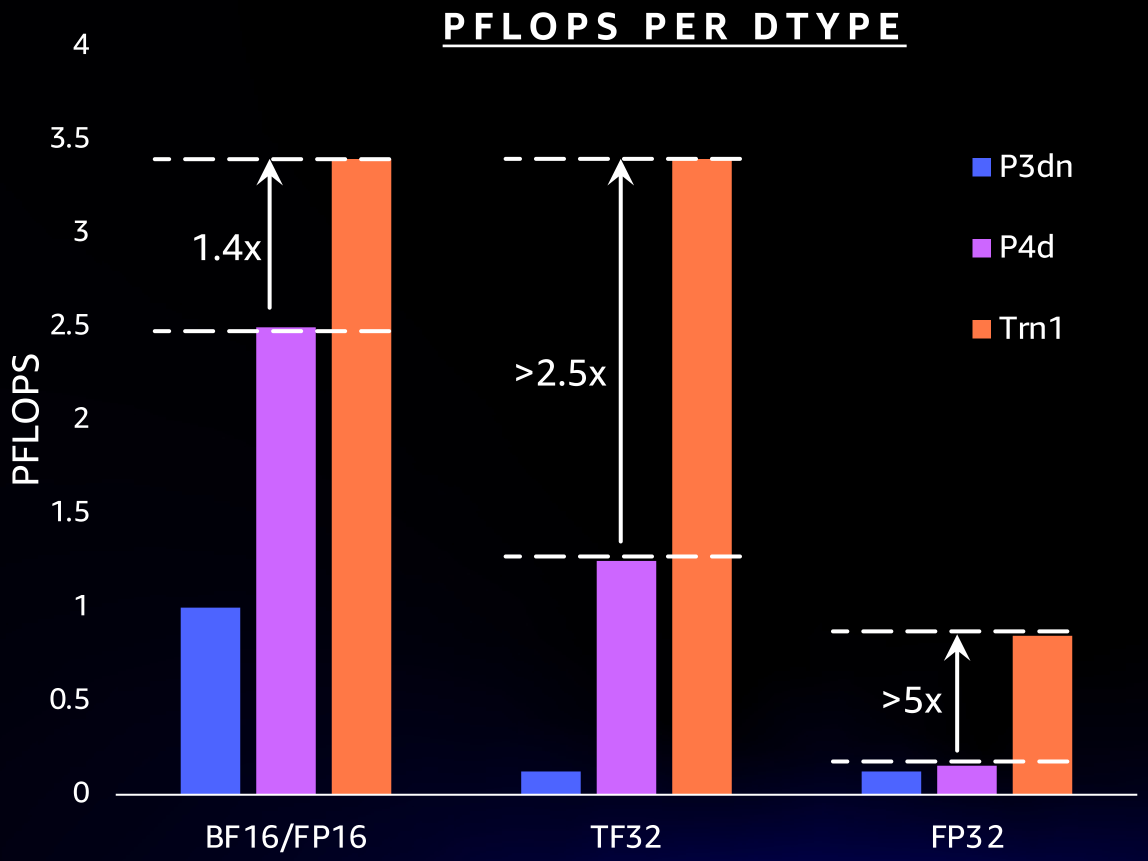
- Support for custom operators



AWS Trainium

- Rich data-type selection

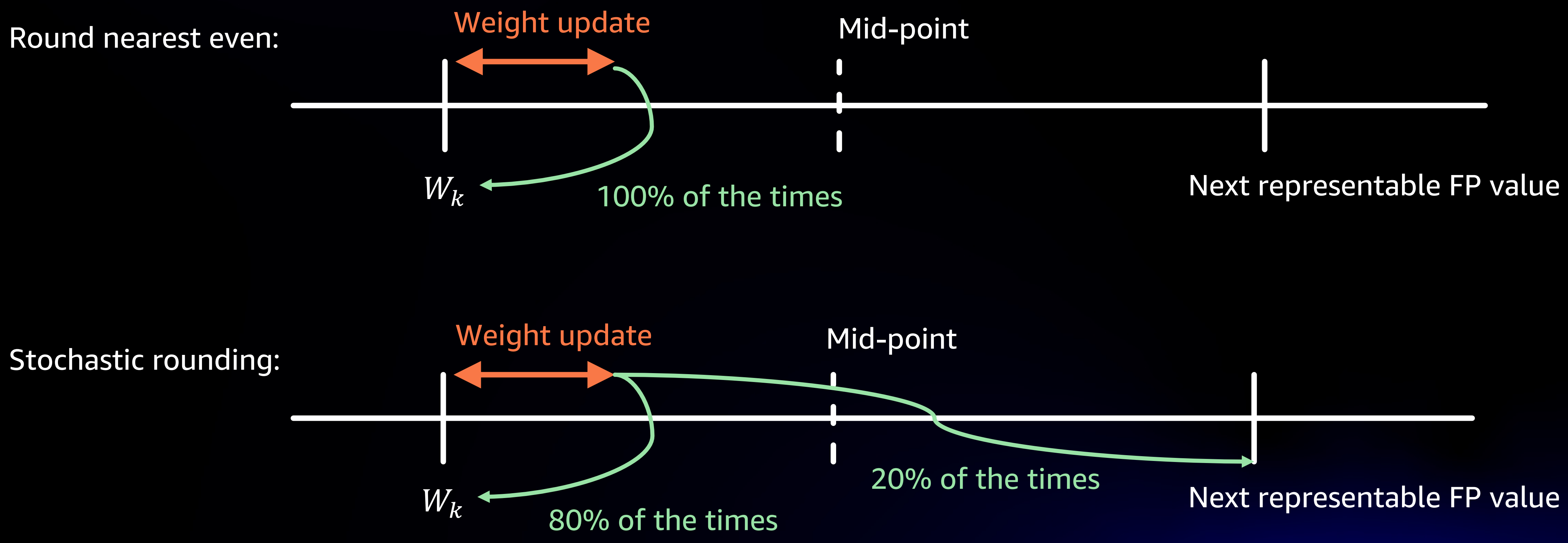




aws

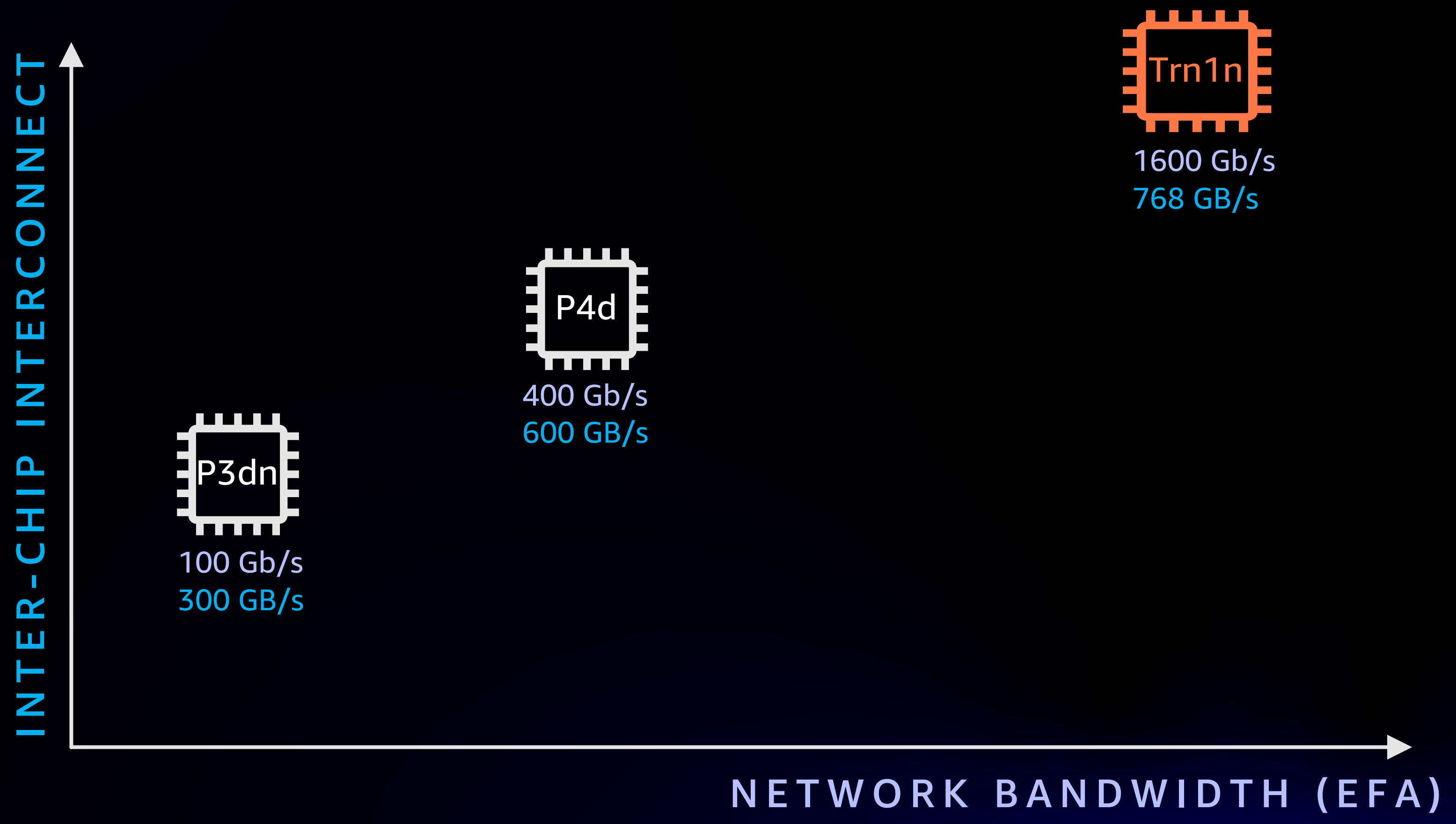
AWS Trainium

- Rich data-type selection
- Stochastic rounding



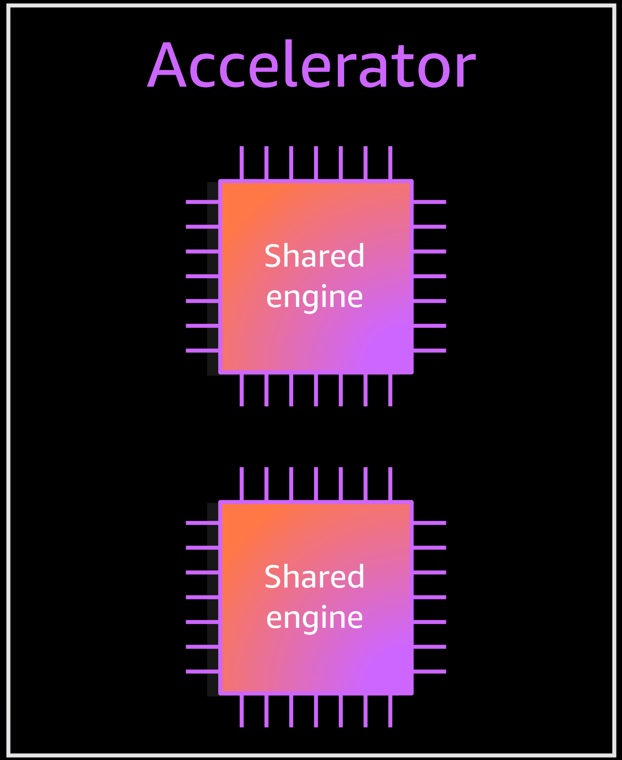
AWS Trainium

- Rich data-type selection
- Stochastic rounding
- High bandwidth,
  Low latency interconnect



AWS Trainium

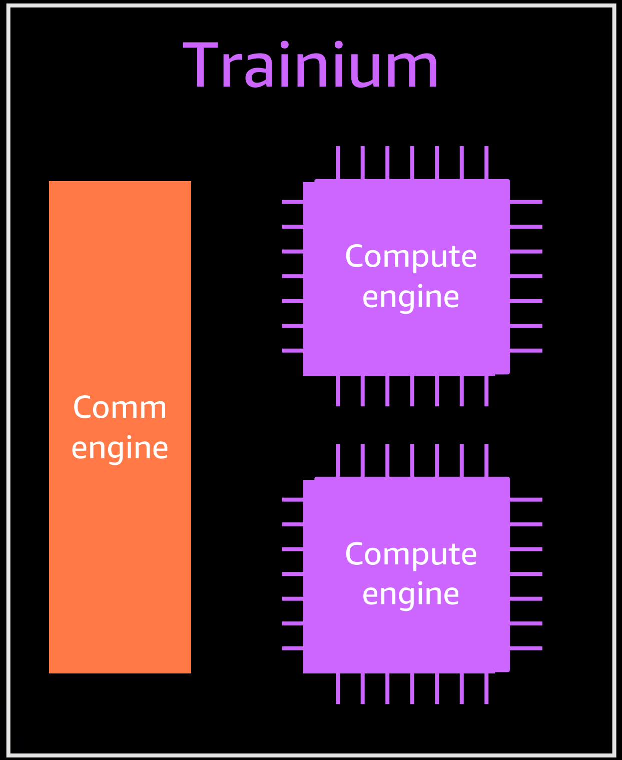
- Rich data-type selection
- Stochastic rounding
- High bandwidth,
  Low latency interconnect
- Parallelized computation and communication

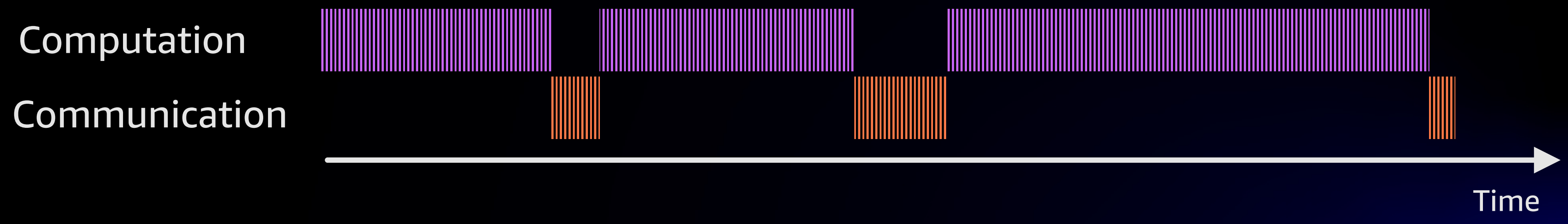


aws

AWS Trainium

- Rich data-type selection
- Stochastic rounding
- High bandwidth,
  Low latency interconnect
- Parallelized computation and communication





AWS Neuron SDK

Supports all major frameworks

PyTorch TensorFlow

Neuron Compiler

Neuron Runtime

Developer tools

AWS Neuron

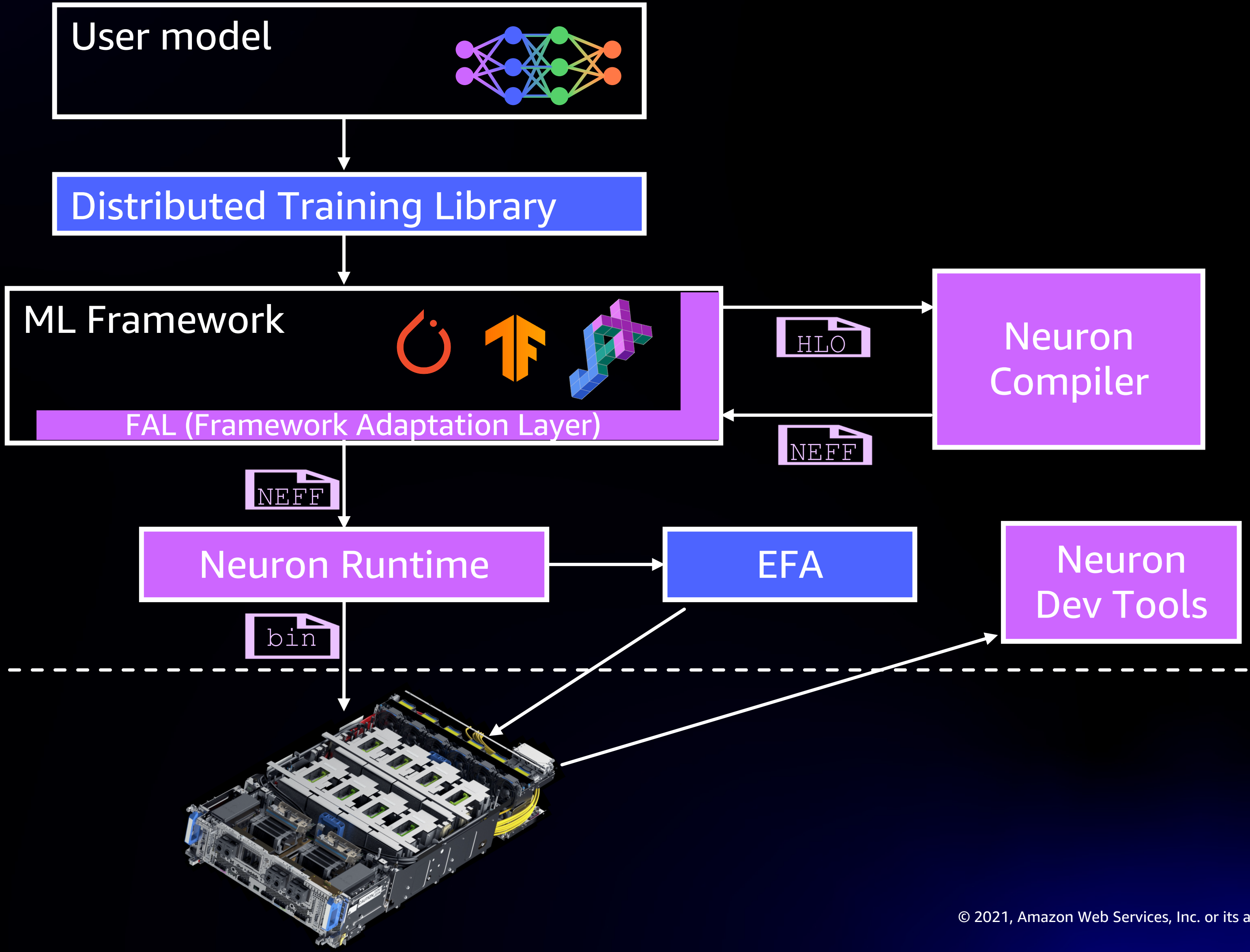
amazon

https://awsdocs-neuron.readthedocs-hosted.com

github.com/aws/aws-neuron-sdk

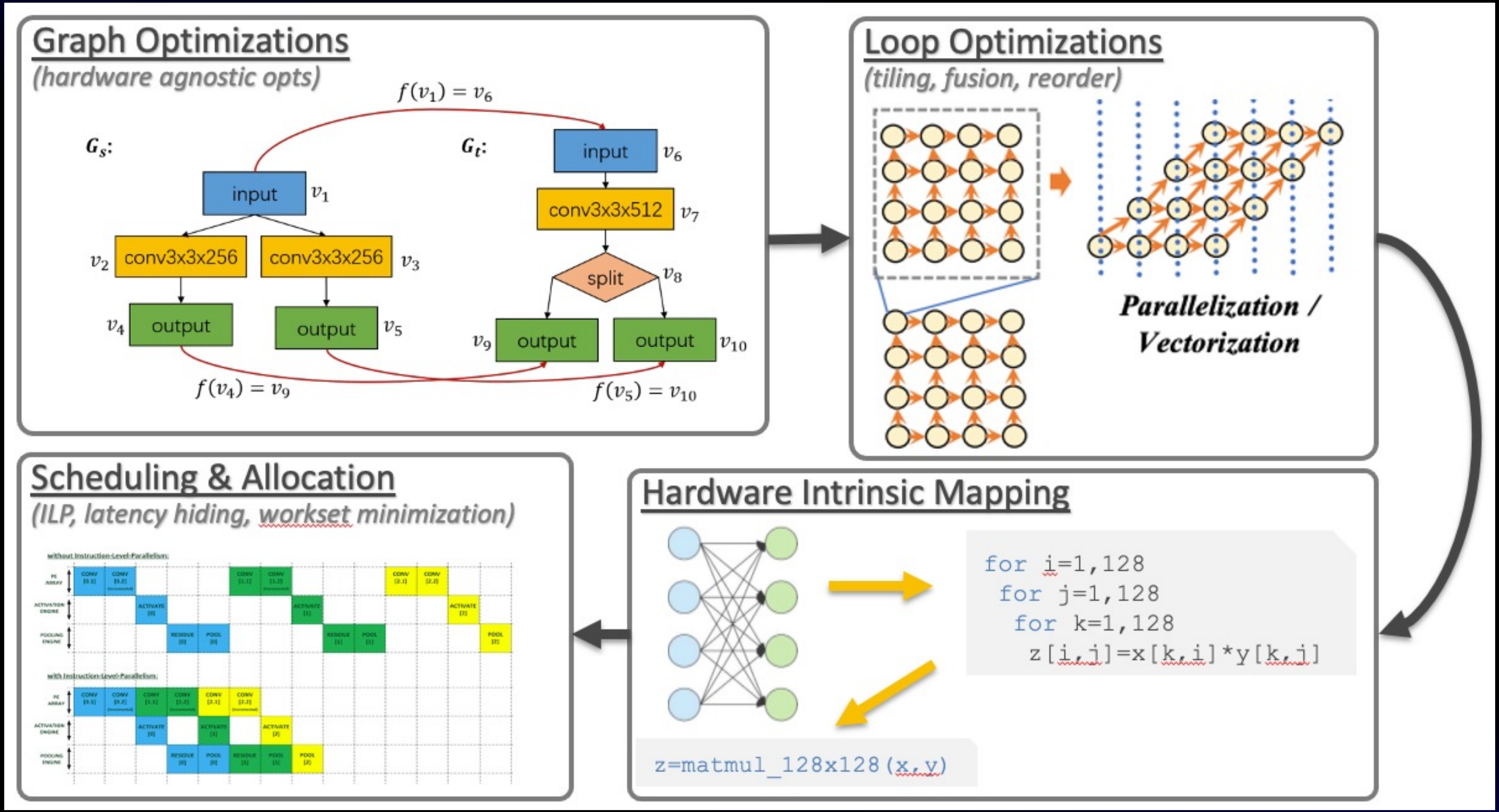
aws

End-to-end flow



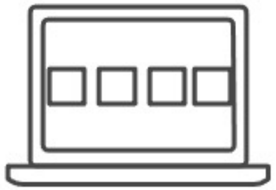
AWS Neuron

AWS Neuron Compiler



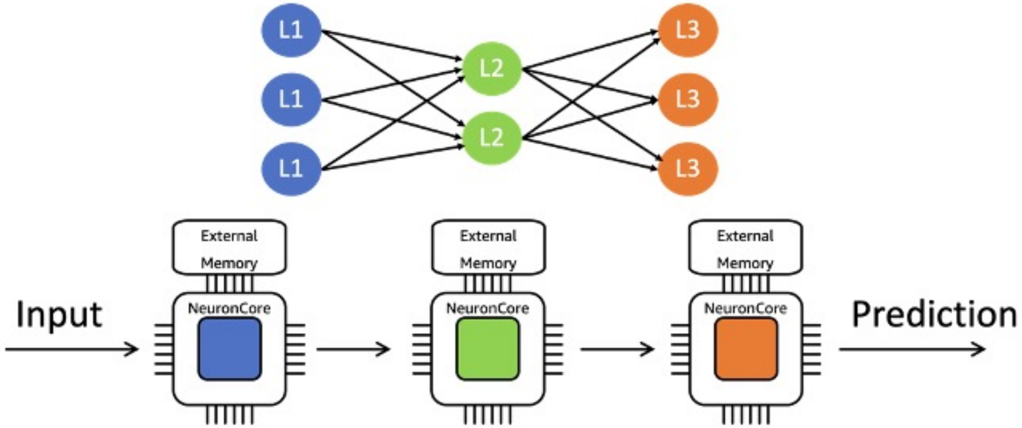
AWS Neuron Runtime

```
ubuntu@ip-172-31-10-131:~$ lspci
...
00:1c.0 System peripheral: Amazon.com, Inc. Device 7064 (rev 01)
00:1d.0 System peripheral: Amazon.com, Inc. Device 7064 (rev 01)
00:1e.0 System peripheral: Amazon.com, Inc. Device 7064 (rev 01)
00:1f.0 System peripheral: Amazon.com, Inc. Device 7064 (rev 01)

ubuntu@ip-172-31-10-131:~$ sudo neuron-ls
+--------------+---------+--------+-----------+-----------+------+------+-----------------------+---------+-------------+
|   PCI BDF    | LOGICAL | NEURON |  MEMORY   |  MEMORY   | EAST | WEST |        RUNTIME        | RUNTIME |  RUNTIME    |
|              |   ID    | CORES  | CHANNEL 0 | CHANNEL 1 |      |      |        ADDRESS        |   PID   |  VERSION    |
+--------------+---------+--------+-----------+-----------+------+------+-----------------------+---------+-------------+
| 0000:00:1c.0 |       0 |      4 | 4096 MB   | 4096 MB   |    1 |    0 | unix:/run/neuron.sock |    6311 | 1.0.7875.0 |
| 0000:00:1d.0 |       1 |      4 | 4096 MB   | 4096 MB   |    1 |    1 | unix:/run/neuron.sock |    6311 | 1.0.7875.0 |
| 0000:00:1e.0 |       2 |      4 | 4096 MB   | 4096 MB   |    1 |    1 | unix:/run/neuron.sock |    6311 | 1.0.7875.0 |
| 0000:00:1f.0 |       3 |      4 | 4096 MB   | 4096 MB   |    0 |    1 | unix:/run/neuron.sock |    6311 | 1.0.7875.0 |
+--------------+---------+--------+-----------+-----------+------+------+-----------------------+---------+-------------+
```



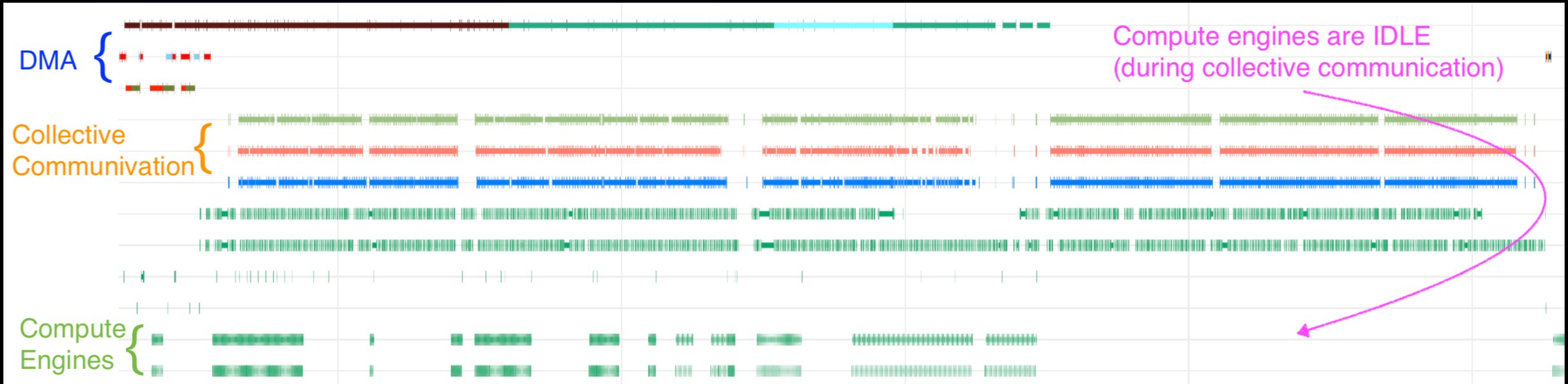
Collective Compute

(Topology + Kernels)



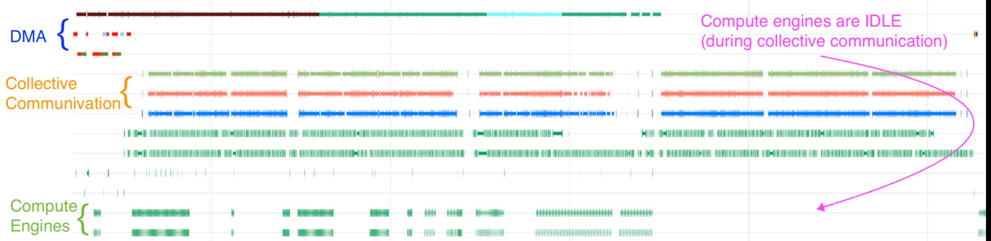
AWS Neuron Profiler

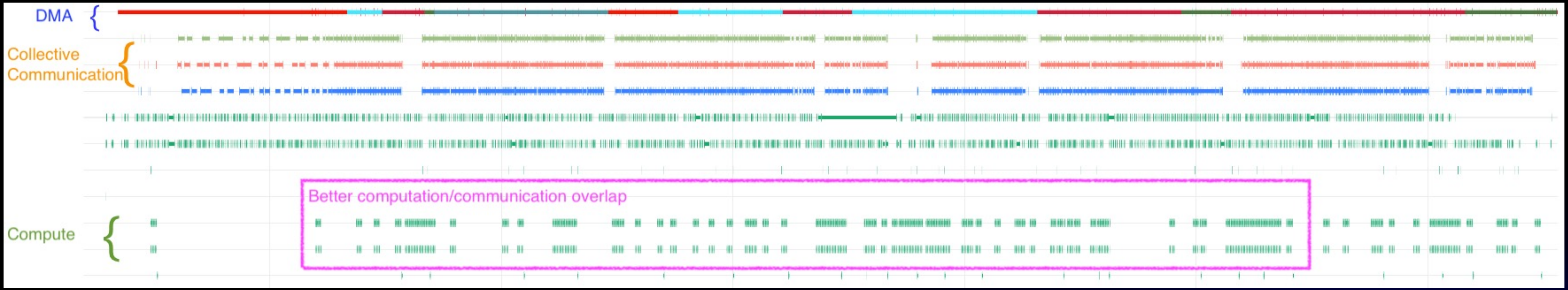
Case study – weight-sharded Transformer:



AWS Neuron Profiler

Case study – weight-sharded Transformer:





AWS Neuron Extensions for Training

AWS Neuron

Framework integration

Full framework integration, JIT, Eager mode, collective compute

Distributed training

Scale up to 10K+ devices, integration with distributed training libraries, and EFA

Flexible and extendable

Support for custom ops, dynamic shapes, new data types, and stochastic rounding

Fully integrated with AWS

SageMaker, EKS, ECS, ParallelCluster, Batch, AMIs

aws

Case study: BERT-Large pre-training

- Bring your own model

```
import os
...
import torch
import torch_xla
import torch_xla.core.xla_model as xm
...
from transformers import BertForPreTraining

model = BertForPreTraining.from_pretrained('bert-large-uncased')

def train_loop_fn(model, optimizer, train_loader, device, epoch, global_step, training_ustep, running_loss):
    max_grad_norm = 1.0
    for i, data in enumerate(train_loader):
        training_ustep += 1
        input_ids, segment_ids, input_mask, masked_lm_labels, next_sentence_labels = data
        outputs = model(input_ids=input_ids,
                        attention_mask=input_mask,
                        token_type_ids=segment_ids,
                        labels=masked_lm_labels,
                        next_sentence_label=next_sentence_labels)
        loss = outputs.loss / flags.grad_accum_usteps
        loss.backward()
        running_loss += loss.detach()

        if (training_ustep + 1) % flags.grad_accum_usteps == 0:
            xm.mark_step()
            running_loss_cpu = running_loss.detach().cpu().item()
            running_loss.zero_()
            torch.nn.utils.clip_grad_norm_(model.parameters(), max_grad_norm)
            xm.optimizer_step(optimizer)
            optimizer.zero_grad()
            scheduler.step()
            global_step += 1
            if global_step >= flags.steps_this_run:
                break

    return global_step, training_ustep, running_loss
```

Case study: BERT-Large pre-training

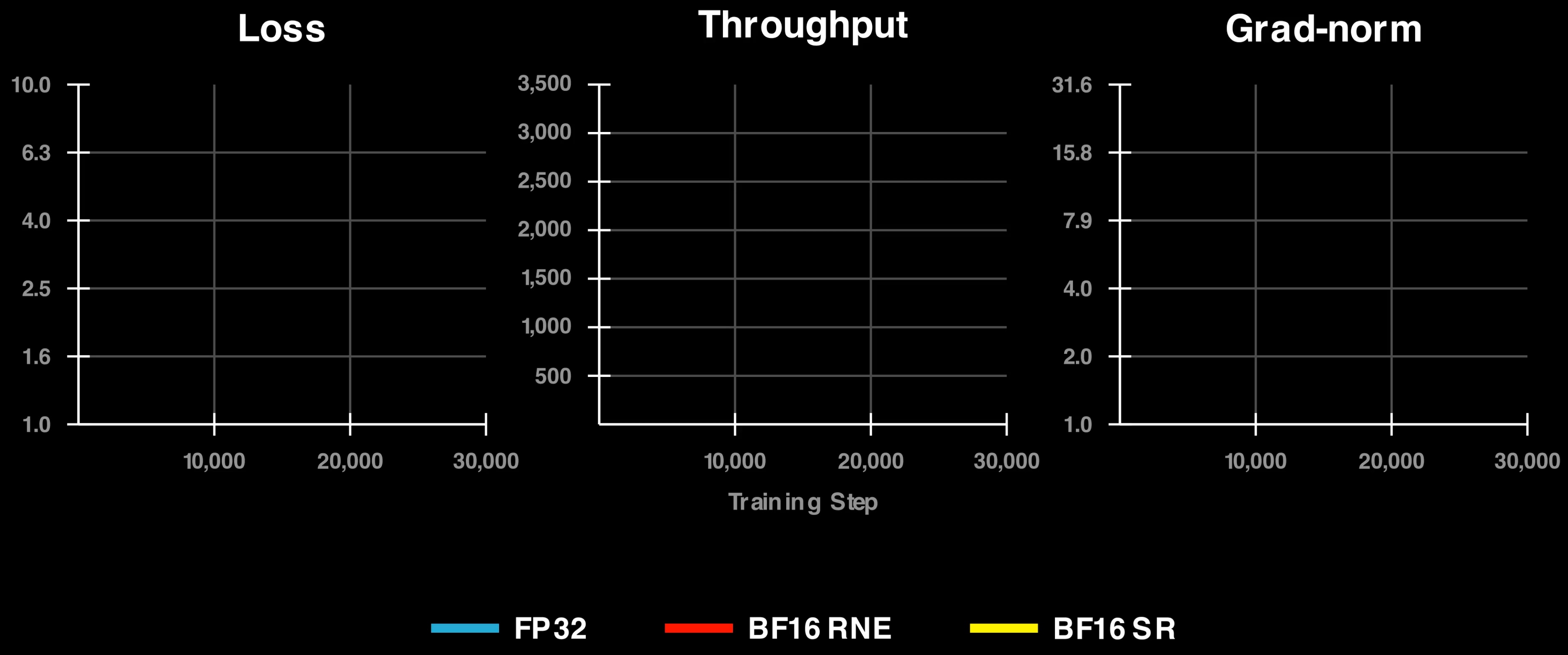
- Bring your own model
- JIT-compile to Trainium

```python
import os
...
import torch
import torch_xla
import torch_xla.core.xla_model as xm
...
from transformers import BertForPreTraining

model = BertForPreTraining.from_pretrained('bert-large-uncased')

def train_loop_fn(model, optimizer, train_loader, device, epoch, global_step, training_ustep, running_loss):
    max_grad_norm = 1.0
    for i, data in enumerate(train_loader):
        training_ustep += 1
        input_ids, segment_ids, input_mask, masked_lm_labels, next_sentence_labels = data
        outputs = model(input_ids=input_ids,
                        attention_mask=input_mask,
                        token_type_ids=segment_ids,
                        labels=masked_lm_labels,
                        next_sentence_label=next_sentence_labels)
        loss = outputs.loss / flags.grad_accum_usteps
        loss.backward()
        running_loss += loss.detach()

        if (training_ustep + 1) % flags.grad_accum_usteps == 0:
            xm.mark_step()
            running_loss_cpu = running_loss.detach().cpu().item()
            running_loss.zero_()
            torch.nn.utils.clip_grad_norm_(model.parameters(), max_grad_norm)
            xm.optimizer_step(optimizer)
            optimizer.zero_grad()
            scheduler.step()
            global_step += 1
            if global_step >= flags.steps_this_run:
                break

    return global_step, training_ustep, running_loss
```

aws

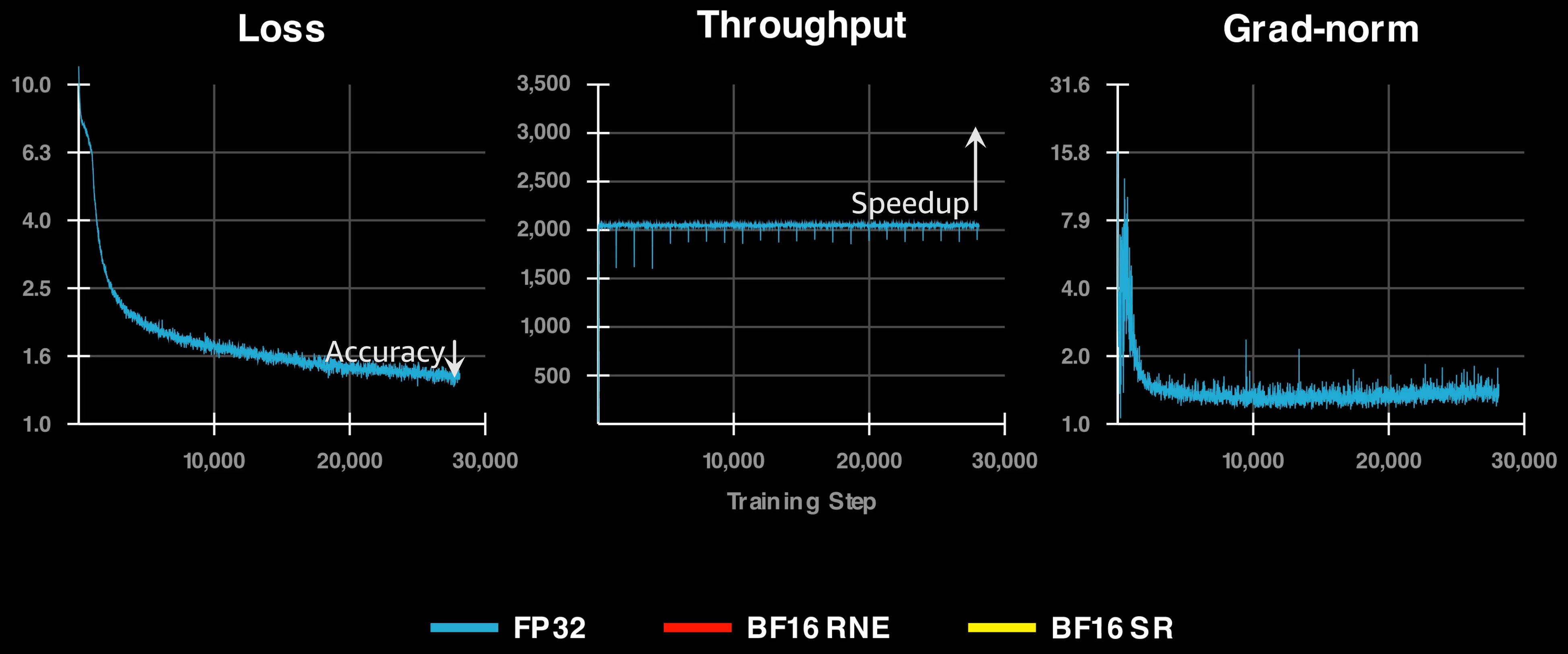
Case study: BERT-Large pre-training

- Bring your own model
- JIT-compile to Trainium
- See it run ☺



Case study: BERT-Large pre-training

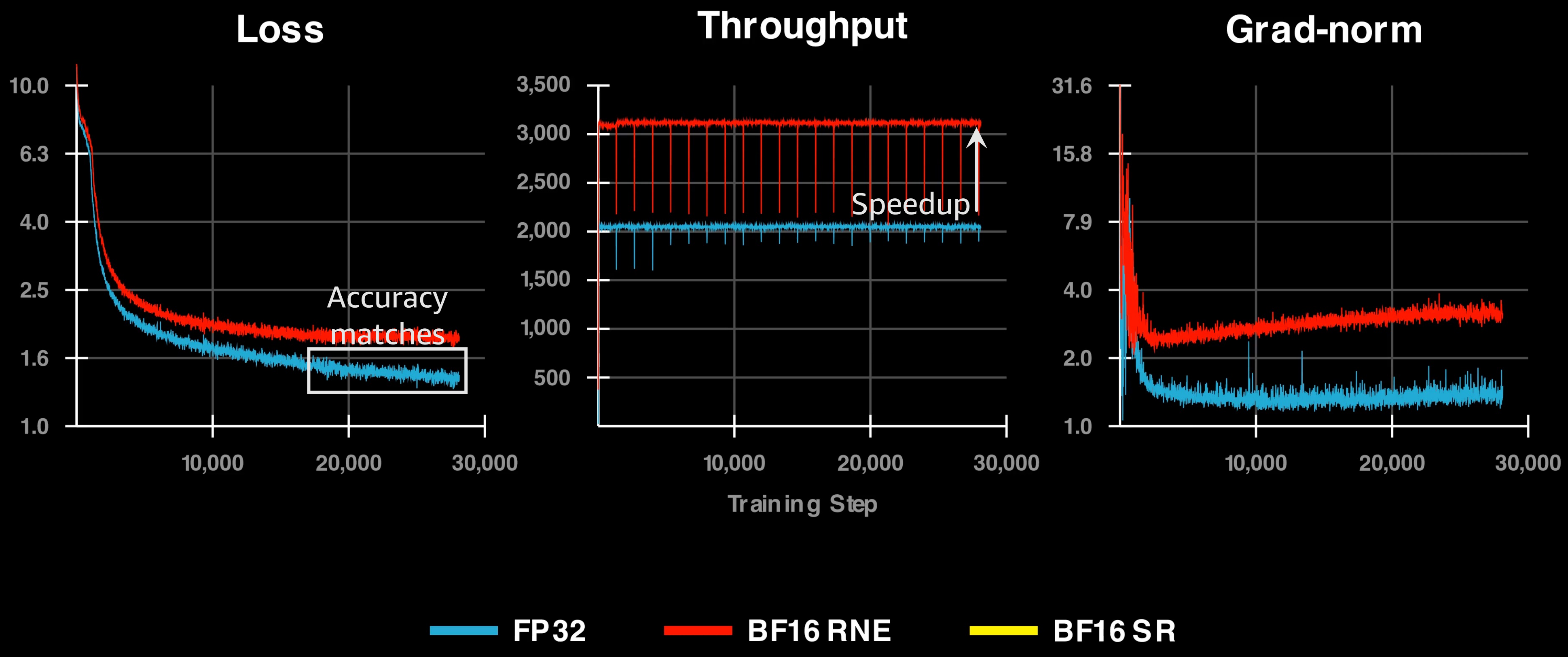
- Bring your own model
- JIT-compile to Trainium
- See it run ☺



aws

Case study: BERT-Large pre-training

- Bring your own model
- JIT-compile to Trainium
- See it run ☺



aws

Amazon EC2 Trn1

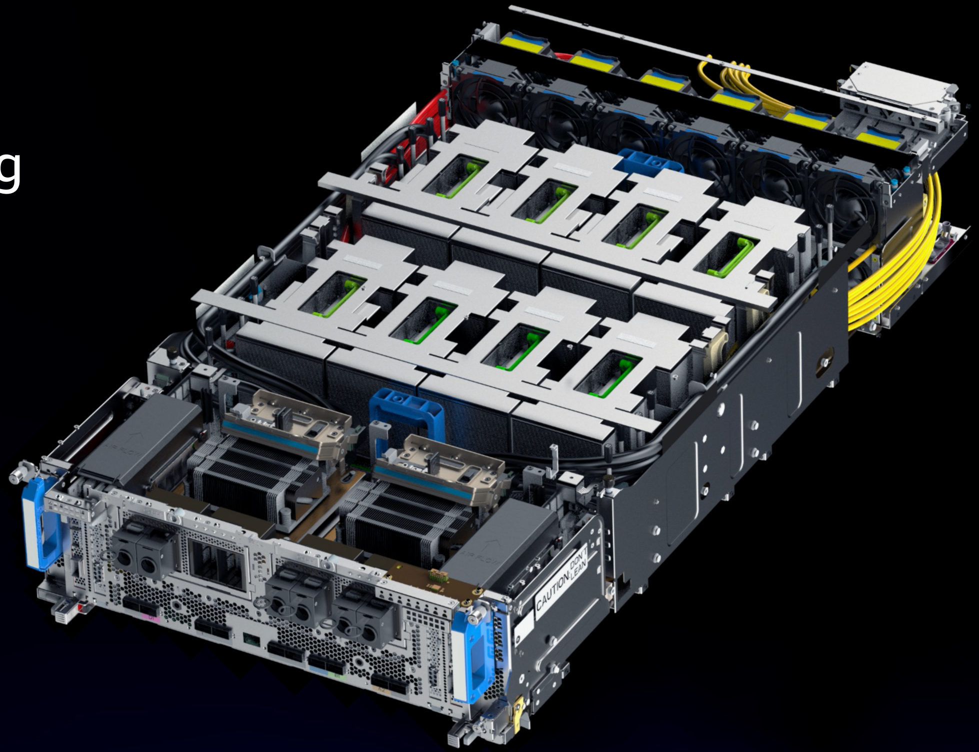
POWERED BY AWS TRAINIUM

Purposely built for the **most cost-efficient DL training in the cloud** for a broad spectrum of applications

AWS is **innovating across the chips, servers, and data center** layers to provide end users with access to cutting edge hardware on-demand

Max developer efficiency with Neuron SDK providing full integration into PyTorch and TensorFlow

Seamless integration with AWS services like SageMaker, Amazon ECS, ParallelCluster and more



Thank you!

We’re hiring! https://www.amazon.jobs/en/landing_pages/annapurna%20labs

Tobias Edler von Koch
Ron Diamant

aws

Q&A