Microsoft

Serving DNNs in Real Time at Datacenter Scale with Project Brainwave

Eric Chung, PhD
Senior Research Manager
Azure Silicon Systems Futures

Breakthroughs in deep learning demand real-time AI

Deep neural networks have enabled major advances in machine learning and AI

Computer vision

Language translation

Speech recognition

Question answering

And more…

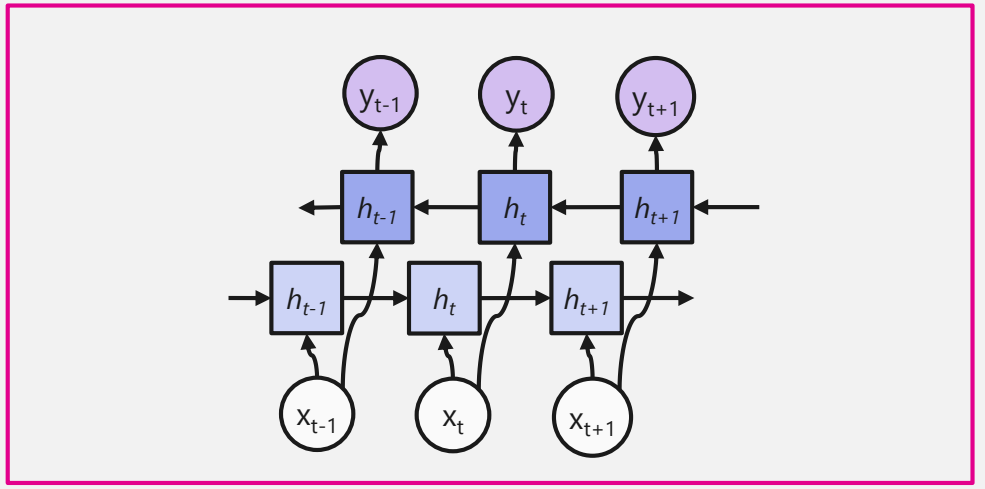
Problem

DNNs are challenging to serve and deploy in large-scale online services

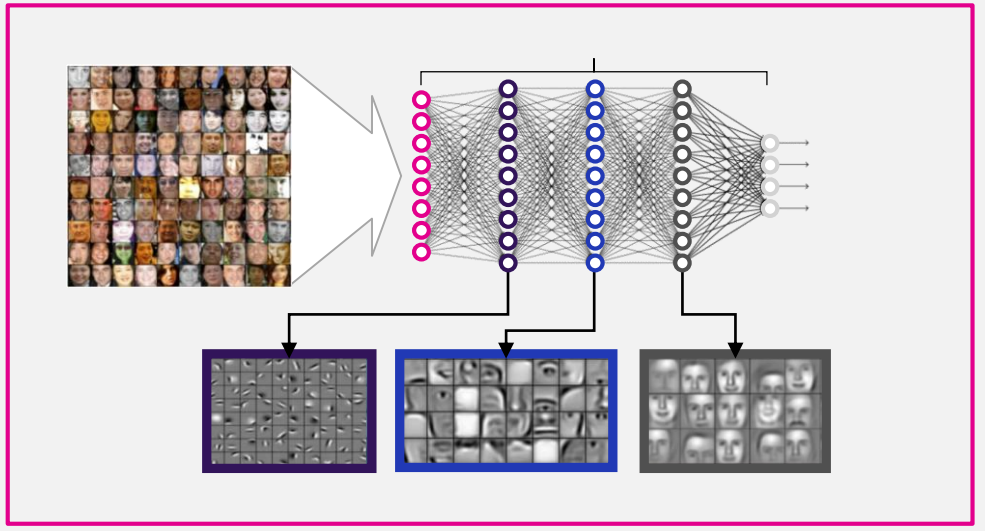
Heavily constrained by latency, cost, and power

Size and complexity outpacing growth of commodity CPUs

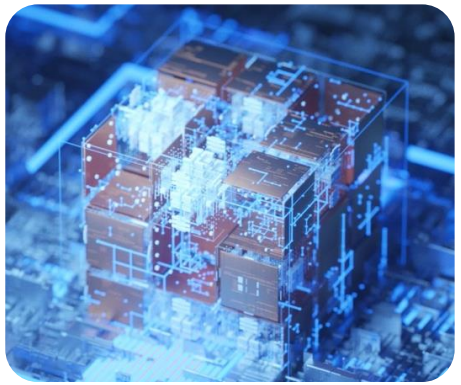
Recurrent Neural Networks



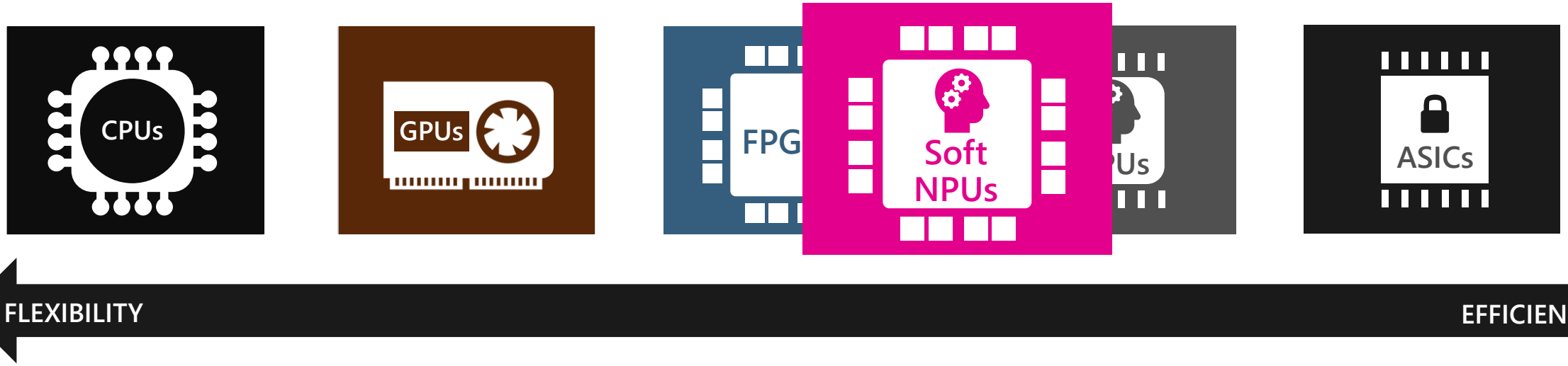
Convolutional Neural Networks

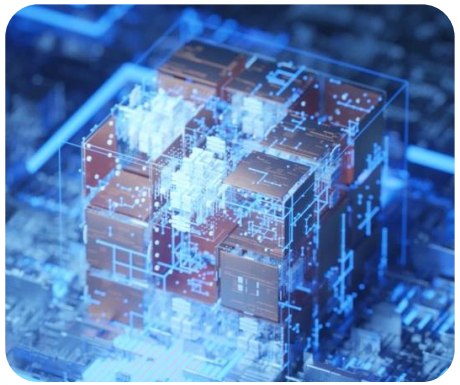


The Rise of Specialized Hardware



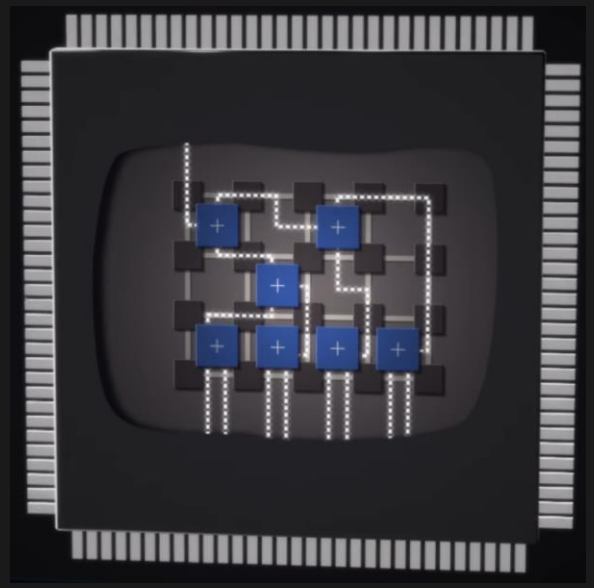
The Rise of Specialized Hardware





History of Microsoft's FPGA Program

Field Programmable Gate Arrays



2011: Project Catapult Launched

2013: Bing pilot runs decision trees 40X faster

2015: Bing ranking throughput increased 2X

2016: Azure Accelerated Networking delivers industry-leading cloud performance

2017: Over 1M servers deployed with FPGAs at hyperscale

2017: Hardware Microservices harness FPGAs for distributed computing

2017: FPGAs enable real-time AI, ultra-low latency inferencing without batching; Bing launches first FPGA-accelerated Deep Neural Network

2018: Project Brainwave launched in Azure Machine Learning

What is FPGA Technology?

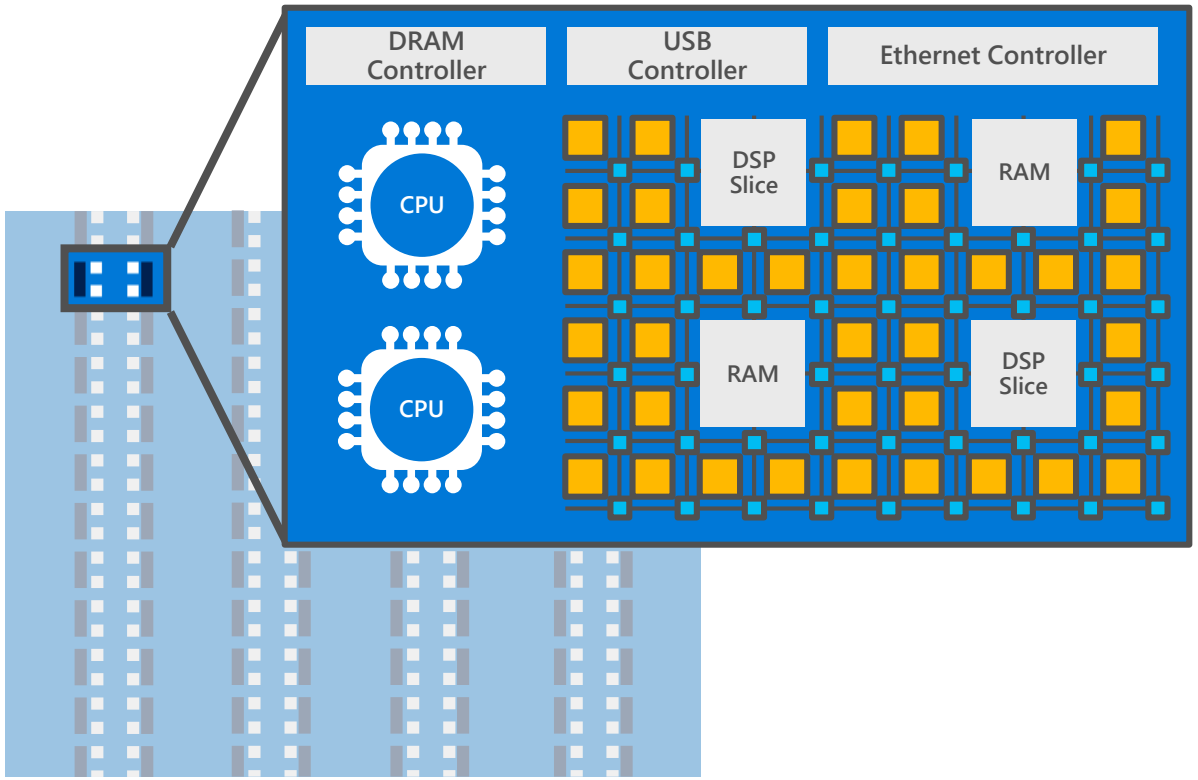
Field Programmable Gate Array

Programmable hardware

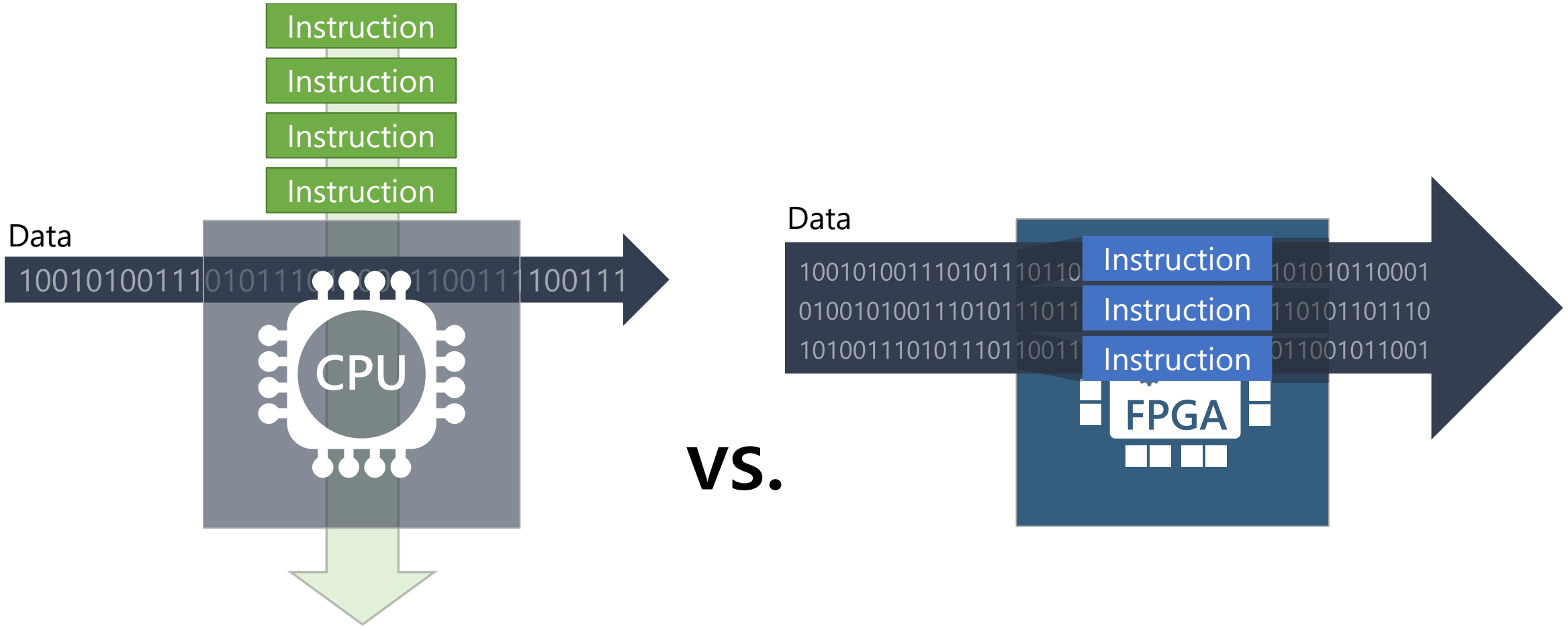
Chip has large quantities of programmable units

Program specialized circuits that communicate directly

FPGA chips are now large SoCs



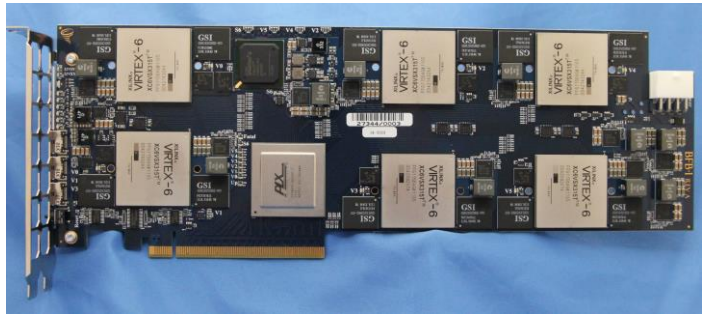
Why FPGAs? Temporal versus Spatial Computing



CPU: temporal compute

FPGA: spatial compute

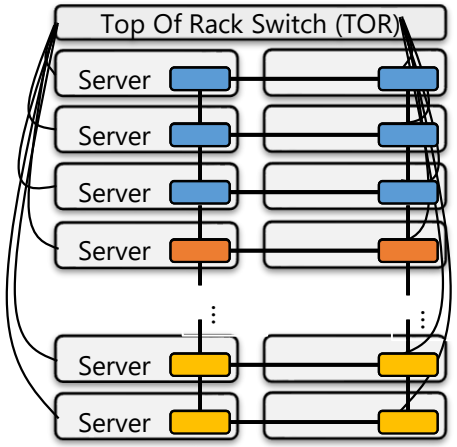
Catapult Hardware Evolution



Gen0

- Complex pipelines fit on single card
- No secondary network

- No elasticity
- Single point of failure
- Slow network to hosts (1Gb)



Gen1

- Homogenous: 1 FPGA per blade
- PCIe access to local FPGA
- Dedicated 6x8 torus allows multi-FPGA accelerators
- Accelerator sharing over torus

- Rack-level elasticity
- Torus adds cost & complexity
- Poor architectural match for infrastructure acceleration

Catapult Gen 2

Ethernet replaces Torus Network

FPGA connected inline *between* NIC and the network switch

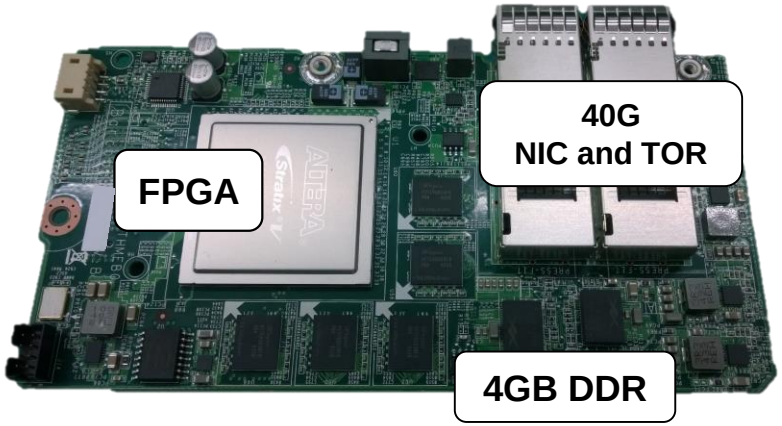
Converged network (FPGA shares datacenter network)

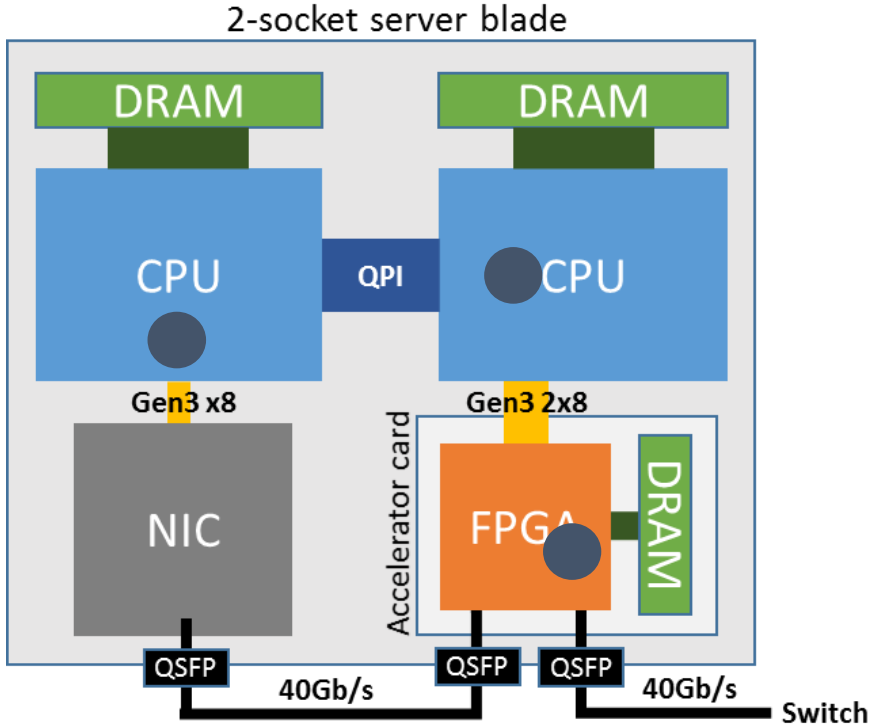
Expands capabilities

Enables infrastructure acceleration, 2x PCIe bandwidth

Architecture justifies economics

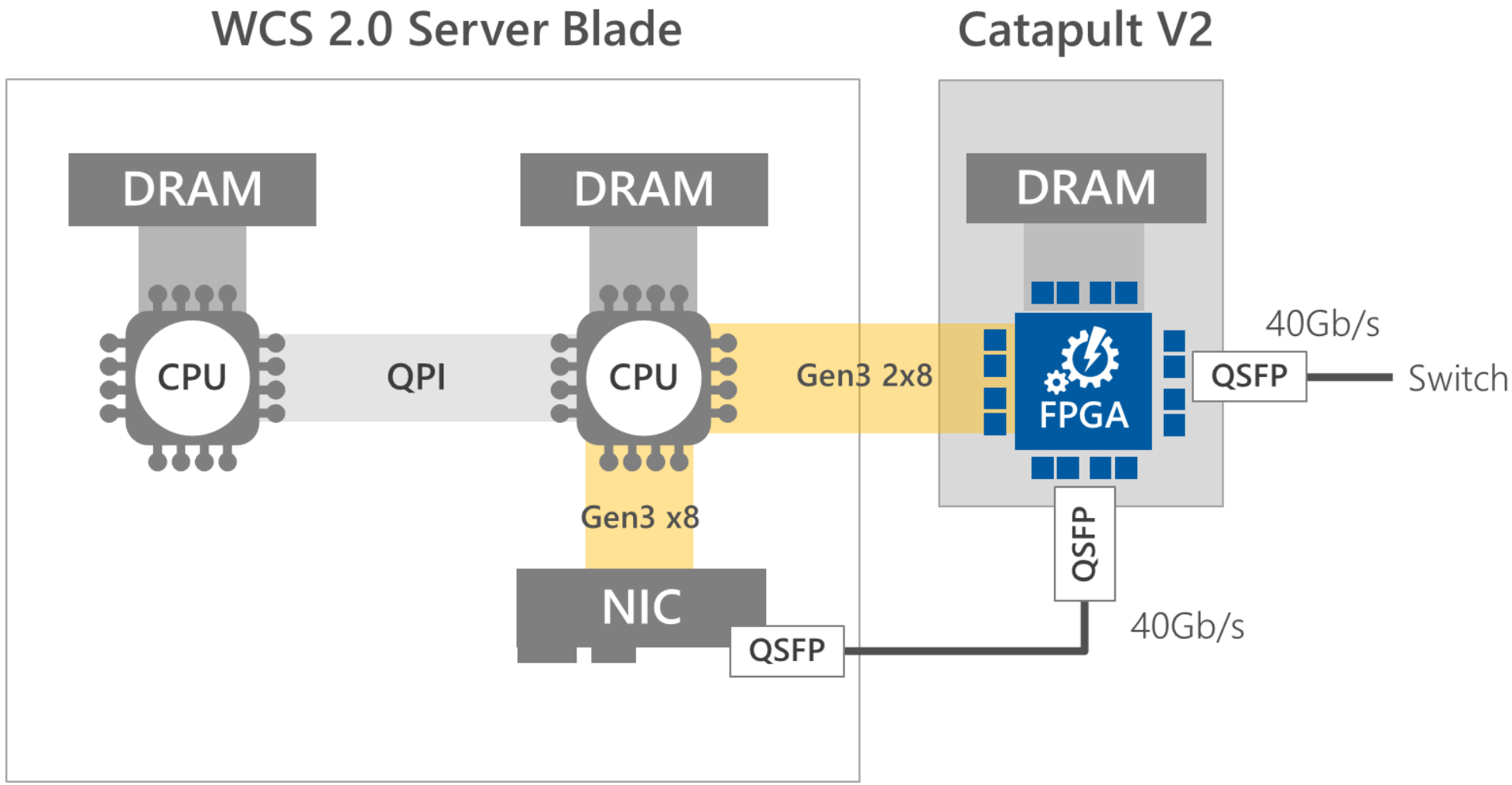
Allow sharing of accelerators over the network to eliminate underutilization





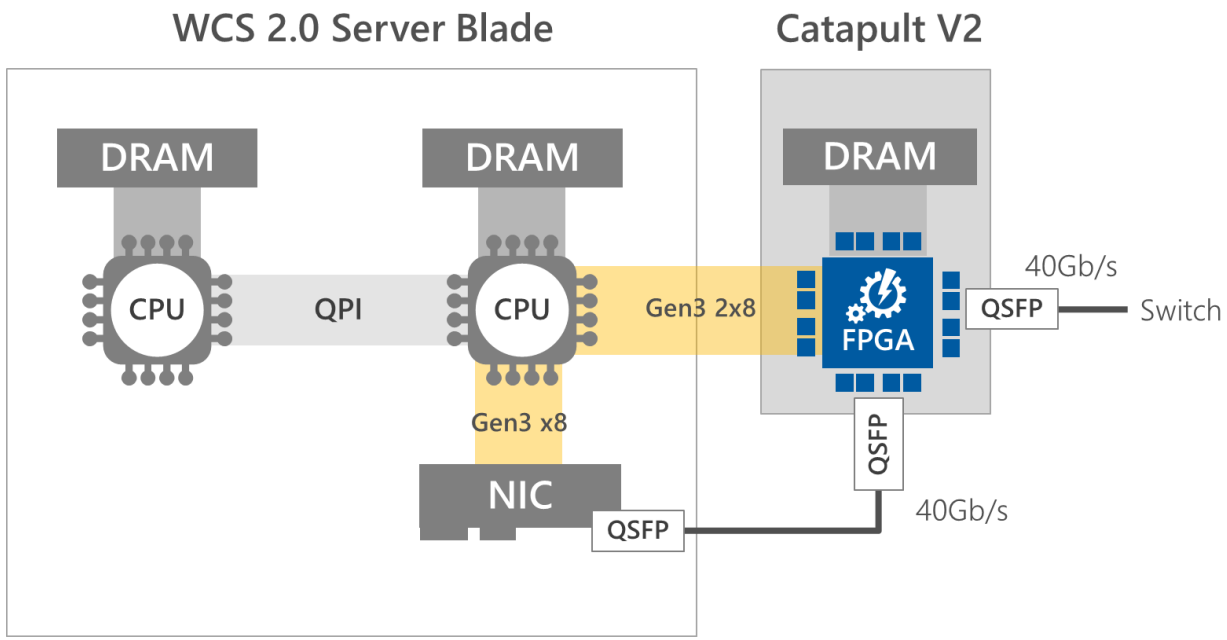
- Local
- Remote
- Infrastructure

Catapult Gen 2 Deployed in MSFT Servers Worldwide



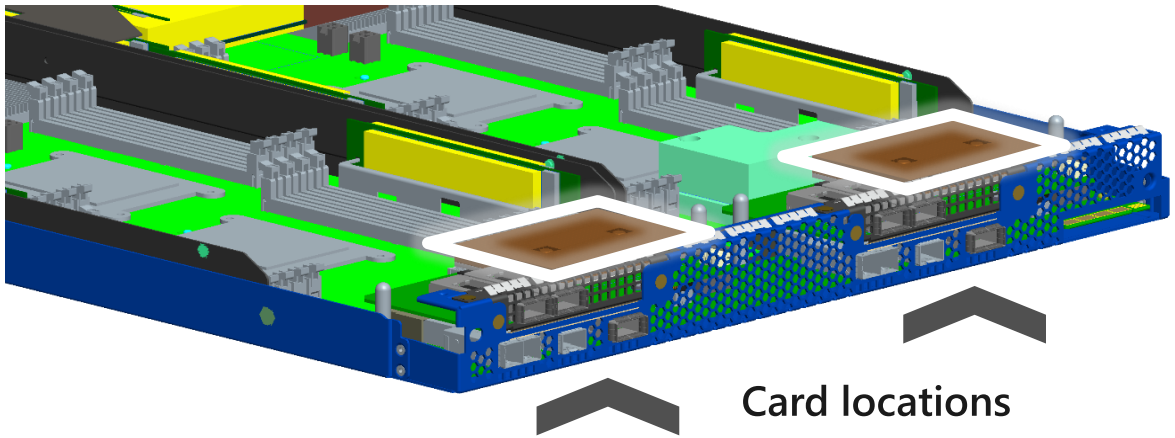
A Cloud-Scale Acceleration Architecture [MICRO'16]

Catapult Gen 2 Deployed in MSFT Servers Worldwide



Catapult v2 Mezzanine card

WCS Gen4.1 Blade with NIC and Catapult FPGA

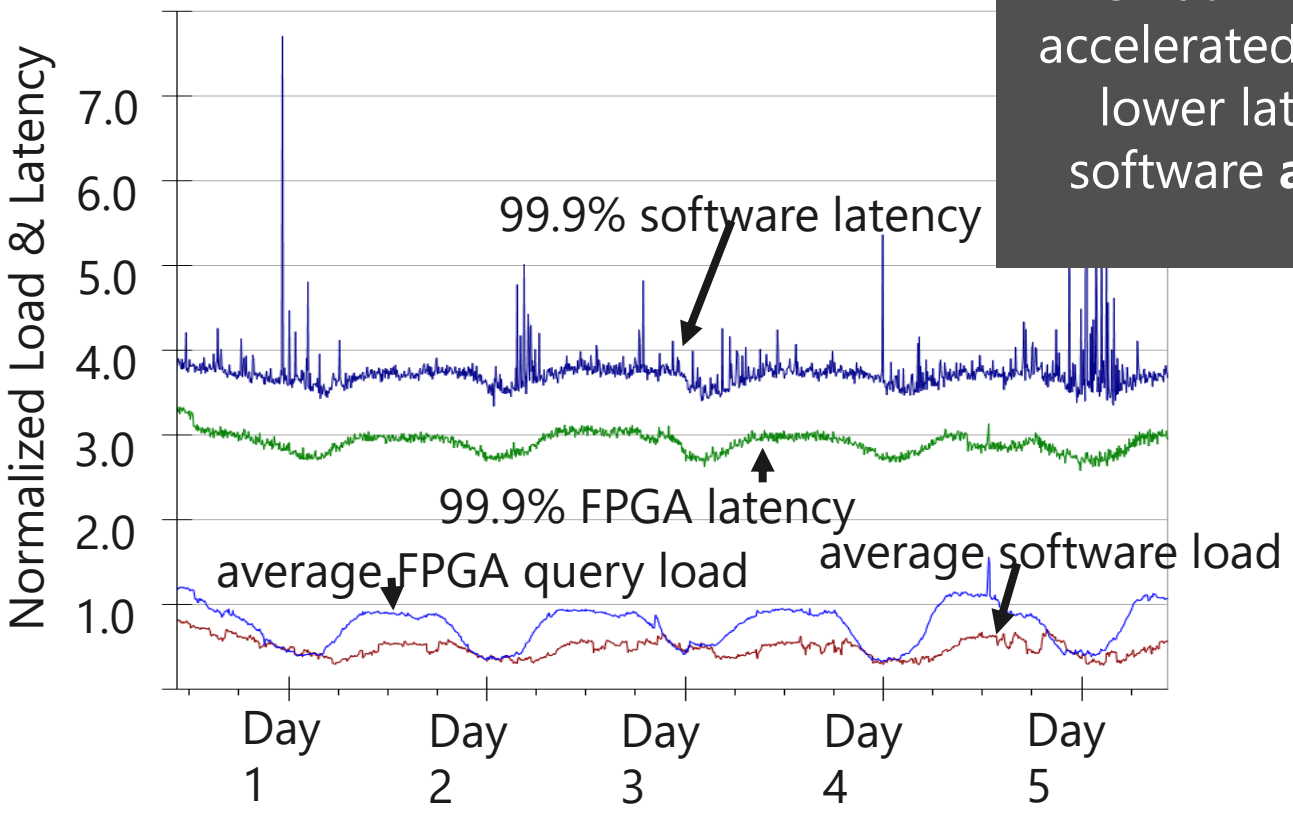


[ISCA'14, HotChips'14, MICRO'16]

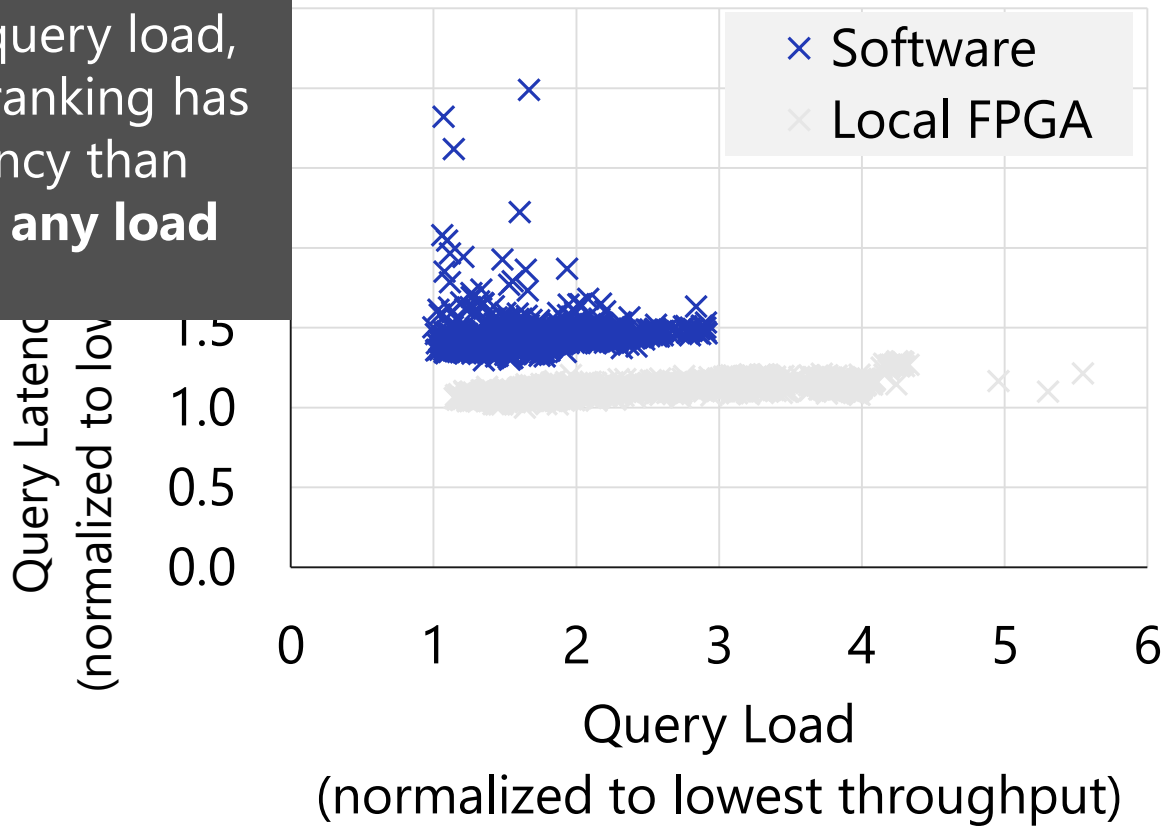
Bing 5 day bed-level latency

Lower & more consistent 99.9th tail latency

In production for 3 years



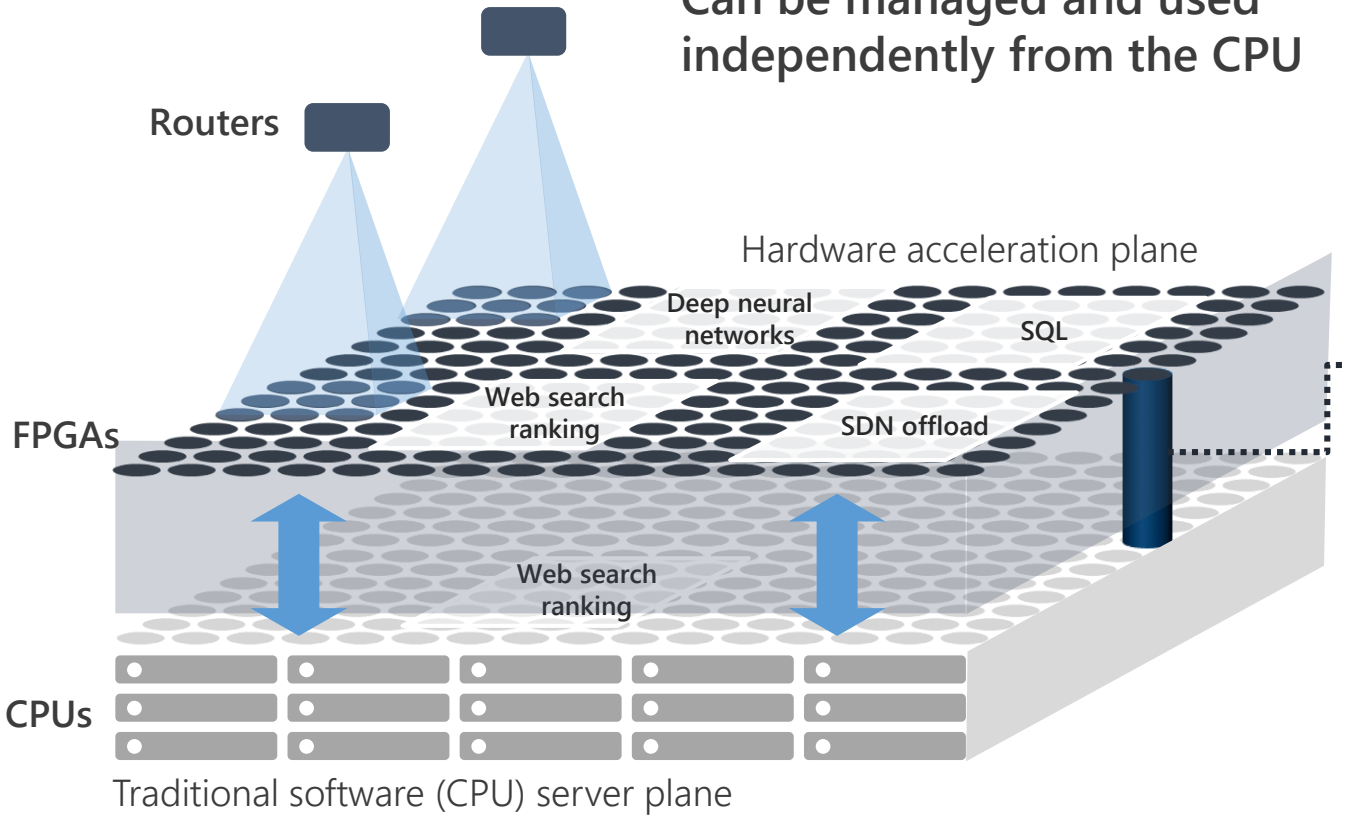
Even at 2× query load, accelerated ranking has lower latency than software **at any load**

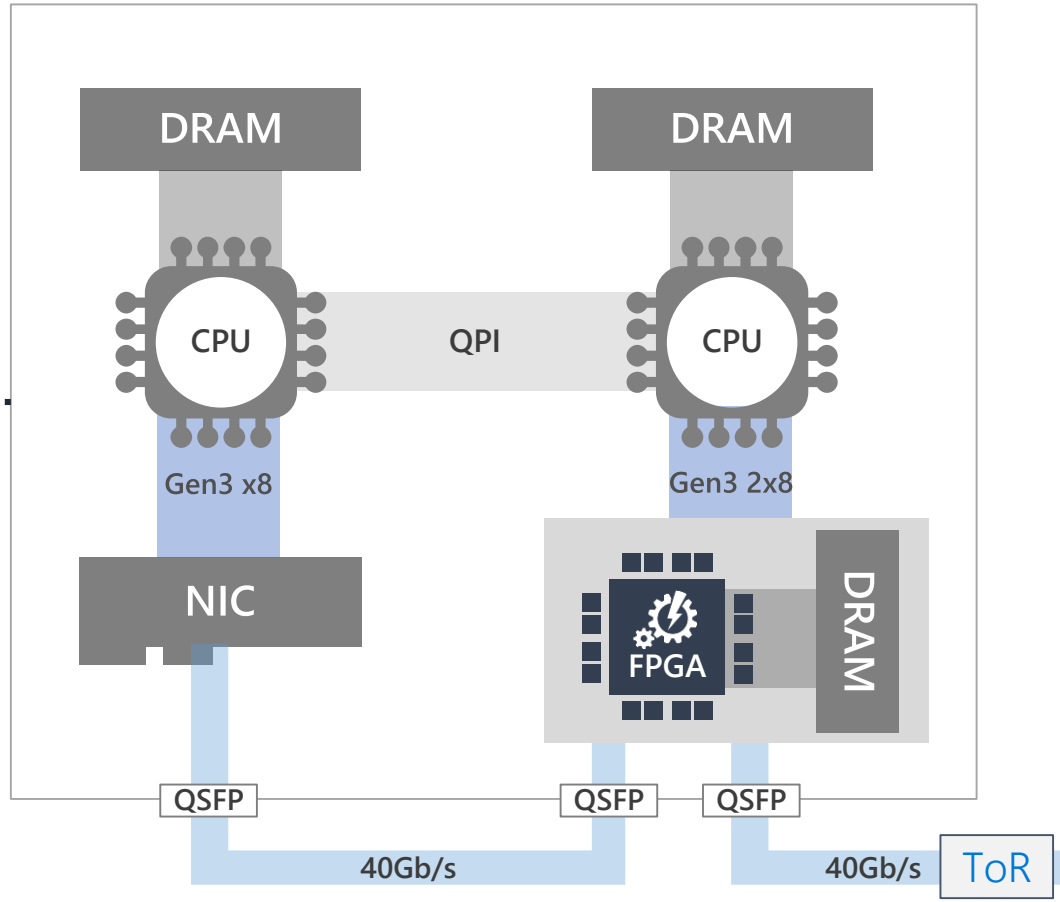


Hardware Microservices on FPGAs

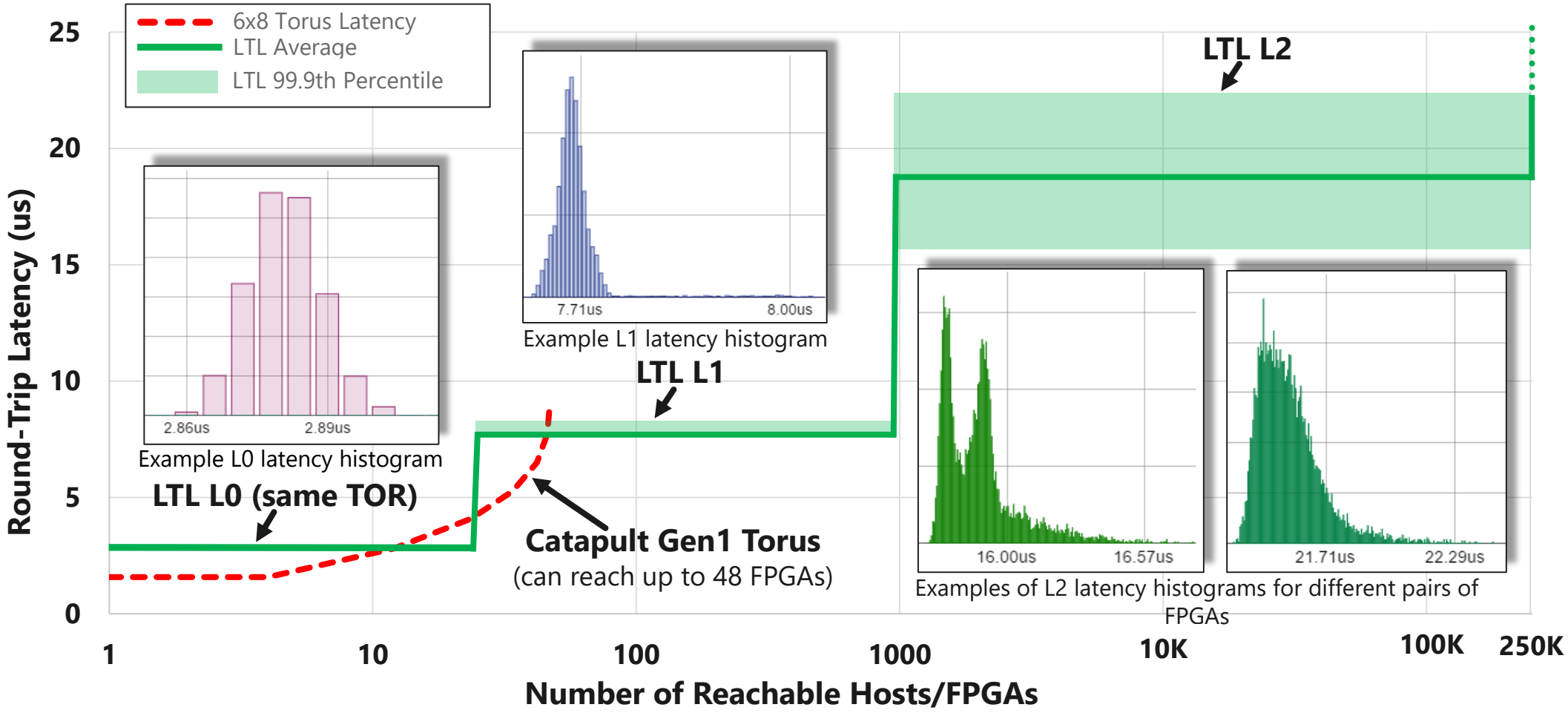
Interconnected FPGAs form a separate plane of computation

Can be managed and used independently from the CPU

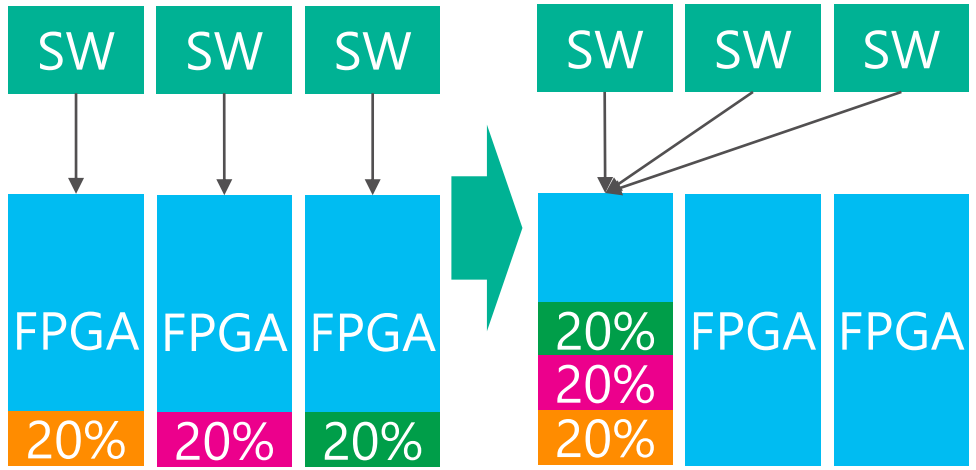




Network reach and latencies



Use case: shared DNN



Economics: consolidation

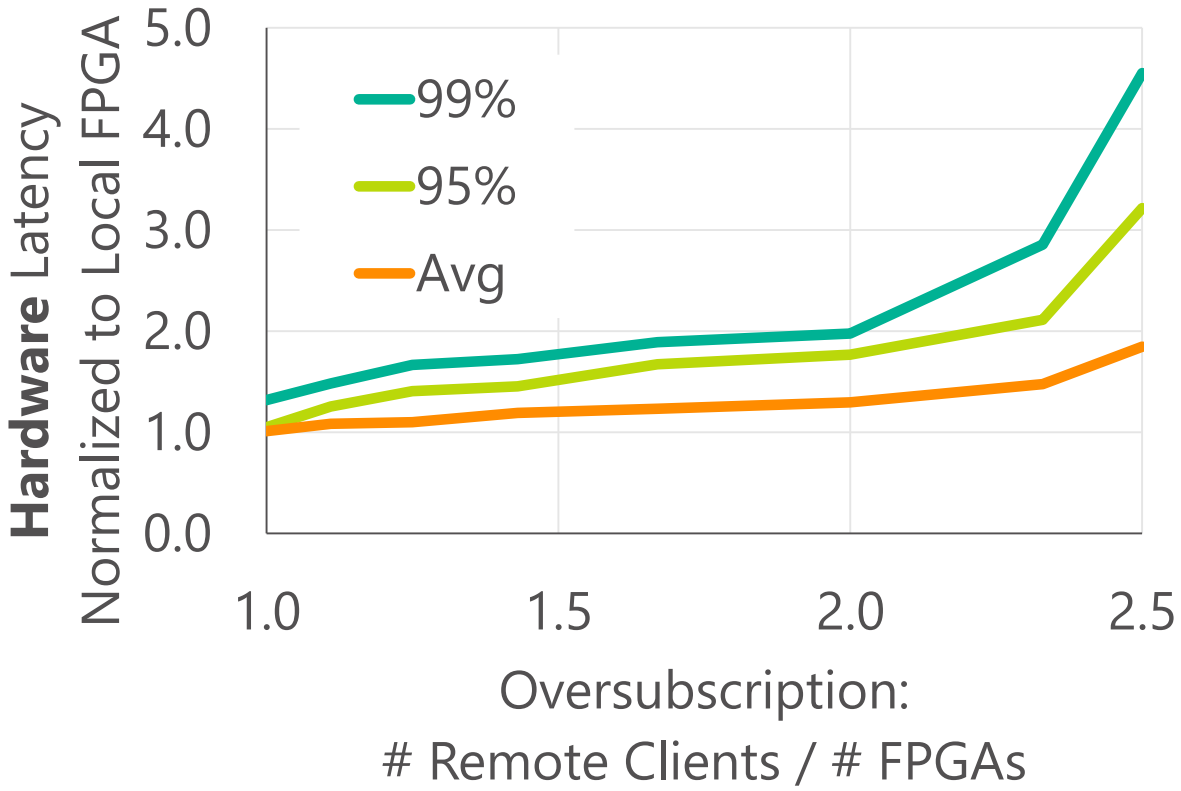
Most accelerators have more throughput than a single host requires

Share excess capacity, use fewer instances

Frees up FPGAs for other use services

DNN accelerator

Sustains 2.5x busy clients in microbenchmark, before queuing delay drives latency up



Microsoft Azure accelerated networking

Software defined networking

Generic Flow Table (GFT) rule based packet rewriting

10x latency reduction vs software, CPU load now <1 core

25Gb/s throughput at 25µs latency – the fastest cloud network

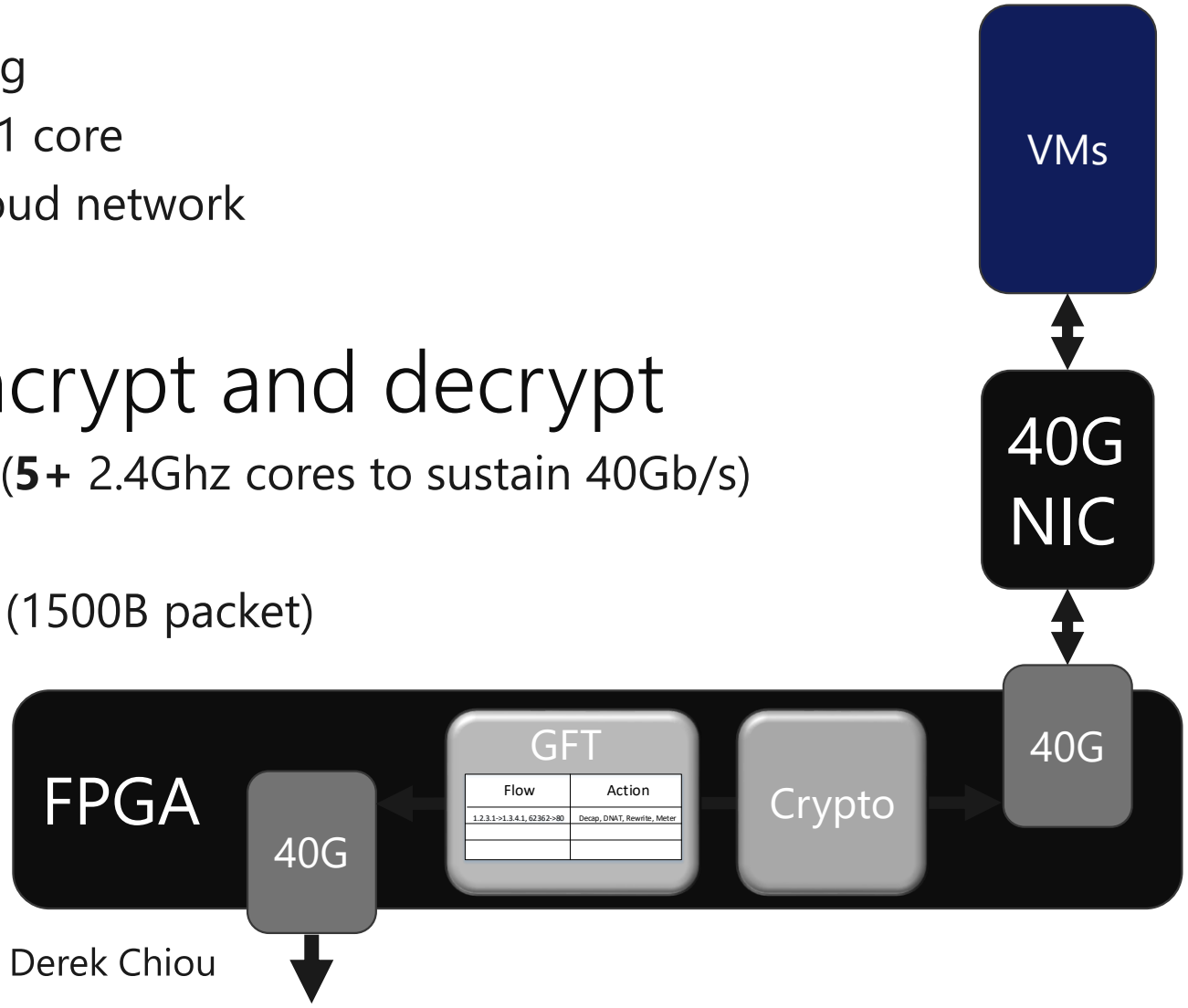
Capable of 40 Gb line rate encrypt and decrypt

On Haswell, AES GCM-128 costs 1.26 cycles/byte[1] (**5+** 2.4Ghz cores to sustain 40Gb/s)

CBC and other algorithms are more expensive

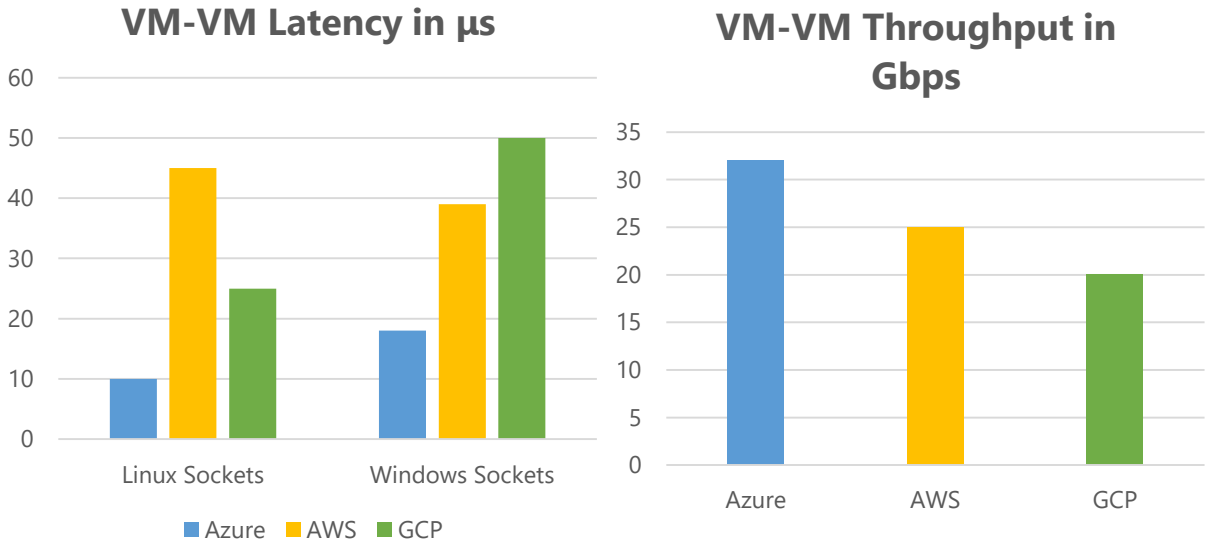
AES CBC-128-SHA1 is 11µs in FPGA vs 4µs on CPU (1500B packet)

Higher latency, but significant CPU savings

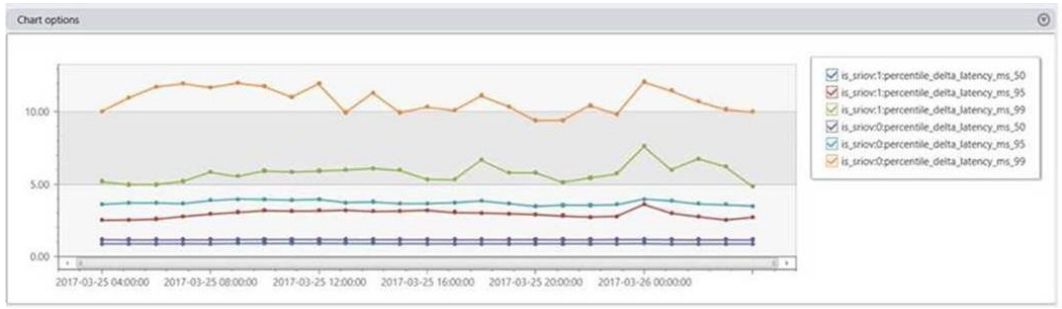


Daniel Firestone, Dave Maltz, Albert Greenberg, Andrew Putnam, Derek Chiou

Microsoft Azure accelerated networking



Azure SQL DB Results on >20k Machines



- 1ms E2E client read times → 300us
- P99 tail write latencies reduced by >2x
- >2x increase in throughput across instances

Customer complaints during rollout - why are some DBs much faster?

Daniel Firestone, Dave Maltz, Albert Greenberg, Andrew Putnam, Derek Chiou

Project Brainwave

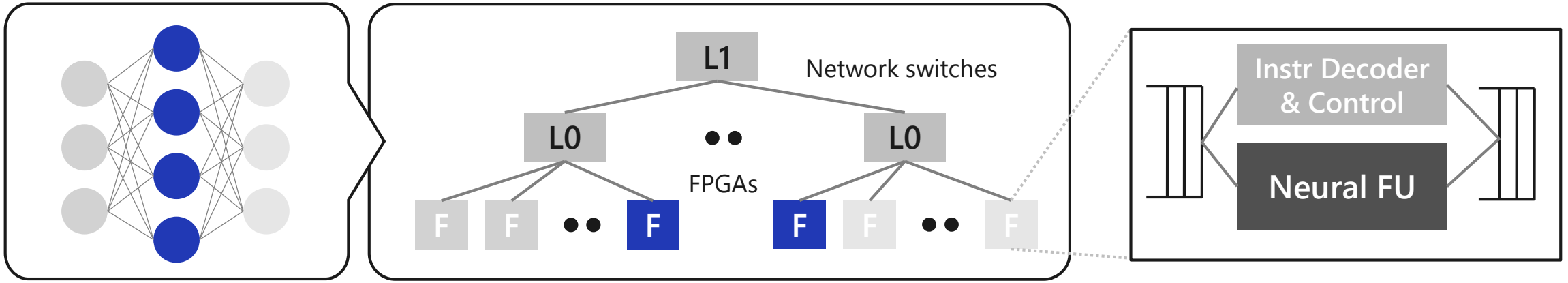
Project Brainwave

A Scalable FPGA-Powered DNN Serving Platform

Fast: Ultra-low latency, high-throughput serving of DNN models at low batch sizes

Flexible: Future proof, adaptable to fast-moving AI space and evolving model types

Friendly: Turnkey deployment of pretrained models



Pretrained DNN Model
in TensorFlow, ONNX, etc.

Scalable DNN Hardware
Microservice

Project Brainwave
Soft NPU

Project Brainwave NPU

High throughput, no batching without sacrificing latency, flexibility

Peak 48 TFLOPS (96,000 MACs) on Intel Stratix 10 FPGAs

Hardware utilization at batch=1 up to 75%

DeepBench RNNs served in under 4 ms, ResNet-50 in under 2 ms

Real-Time NPU Design

Design Tenets for a Real-Time NPU

Objectives

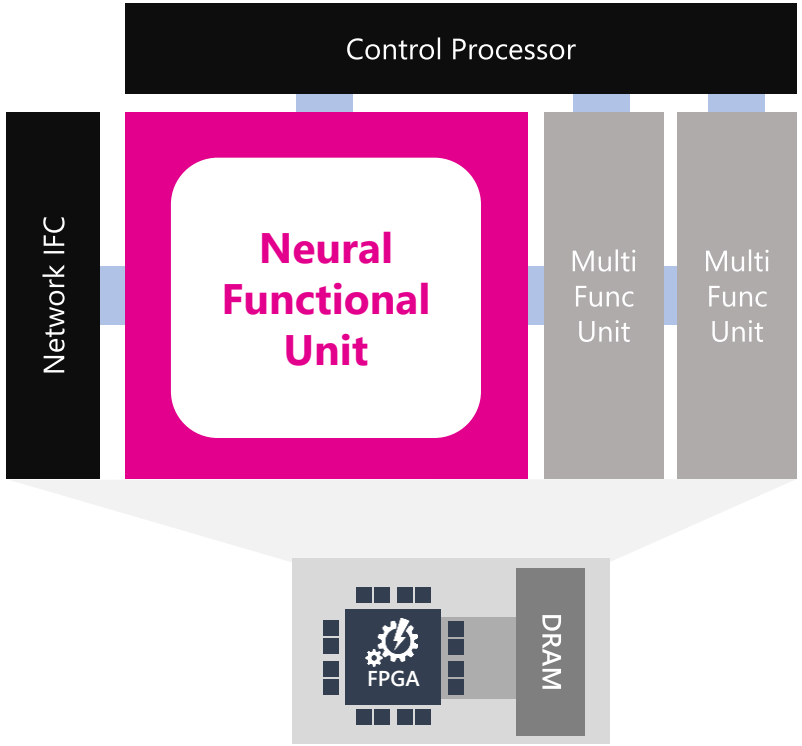
- Optimized for no-batch inferencing
- Simplify programmability and targetability with a single thread of control
- Low complexity implementations

Balance instruction granularity and flexibility

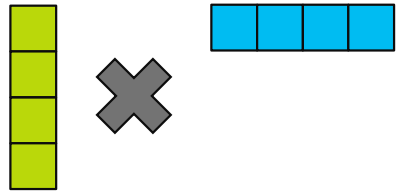
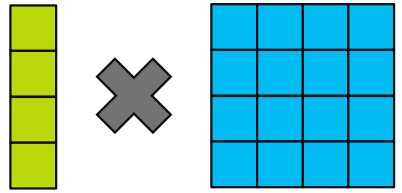
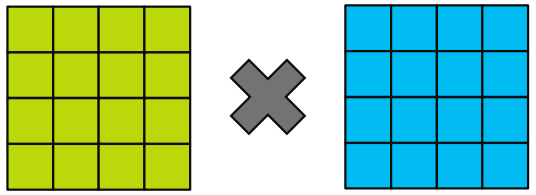
- All instructions operate on multiples of a native dimension N
- Abstracted to M-V and V-V operations
- Composable into diverse models: RNN, 1D+2D CNNs, etc.

Instruction chaining

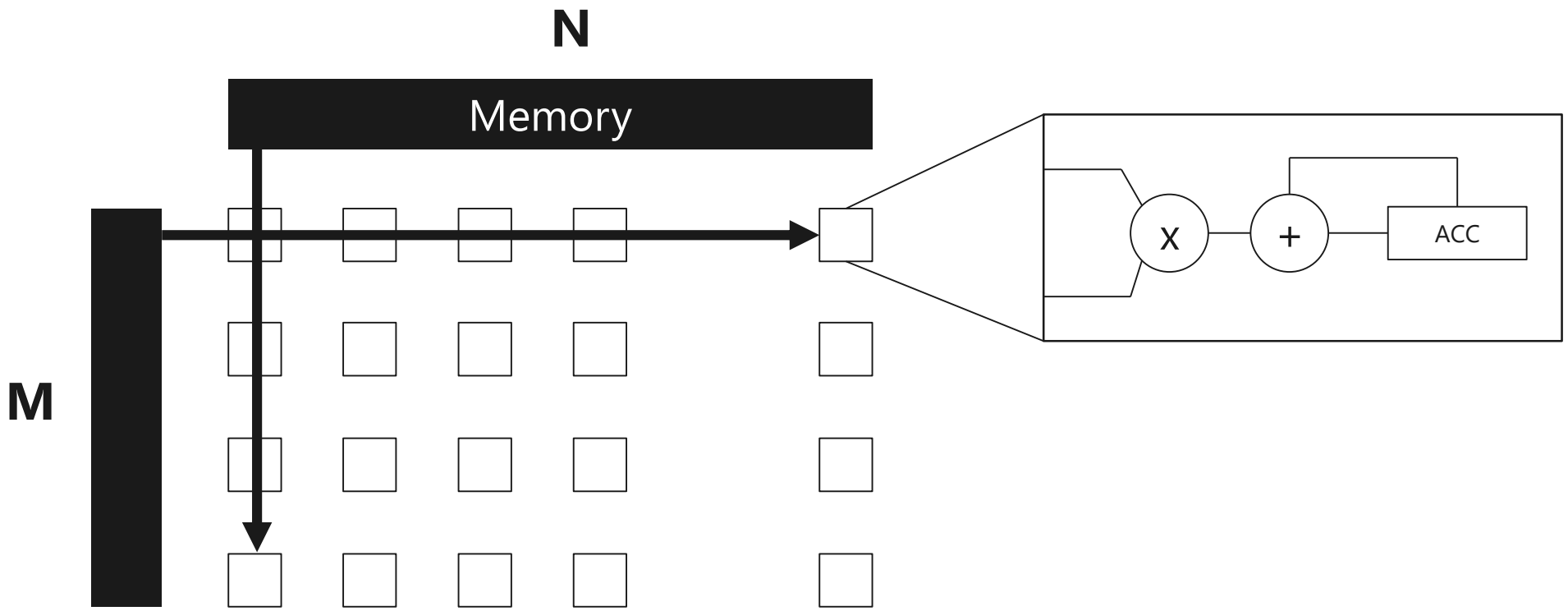
- Instruction results are forwarded to the next instruction
- Enables long chains without dependency analysis or multi-ported register files



Dominant Compute Pattern in ML/DL: Matrix Multiply

Primitive	Good for	Data Reuse	
Vector * Vector	CNNs RNNs FCs	*IN0: O(1)* *IN1: O(1)*	
Matrix * Vector	CNNs RNNs FCs	*IN0: O(1)* *IN1: O(N)*	
Matrix * Matrix	CNNs Batched FCs Batched RNNs	*IN0: O(N)* *IN1: O(N)*	

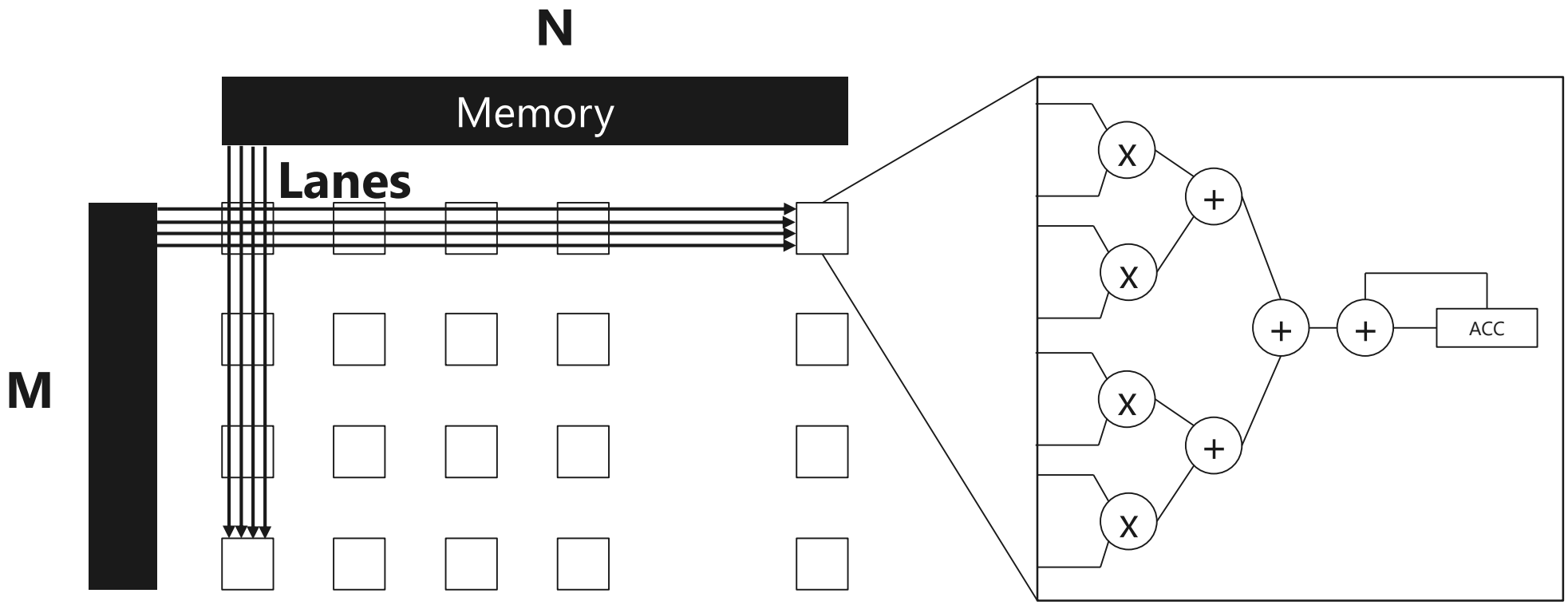
Matrix Multiply on Tensor Arrays



MACs = MN

Read bandwidth = M+N loads / cycle

Matrix Multiply on Vectorized Tensor Arrays



MACs = Lanes x M x N

Read bandwidth = Lanes x (M+N) loads / cycle

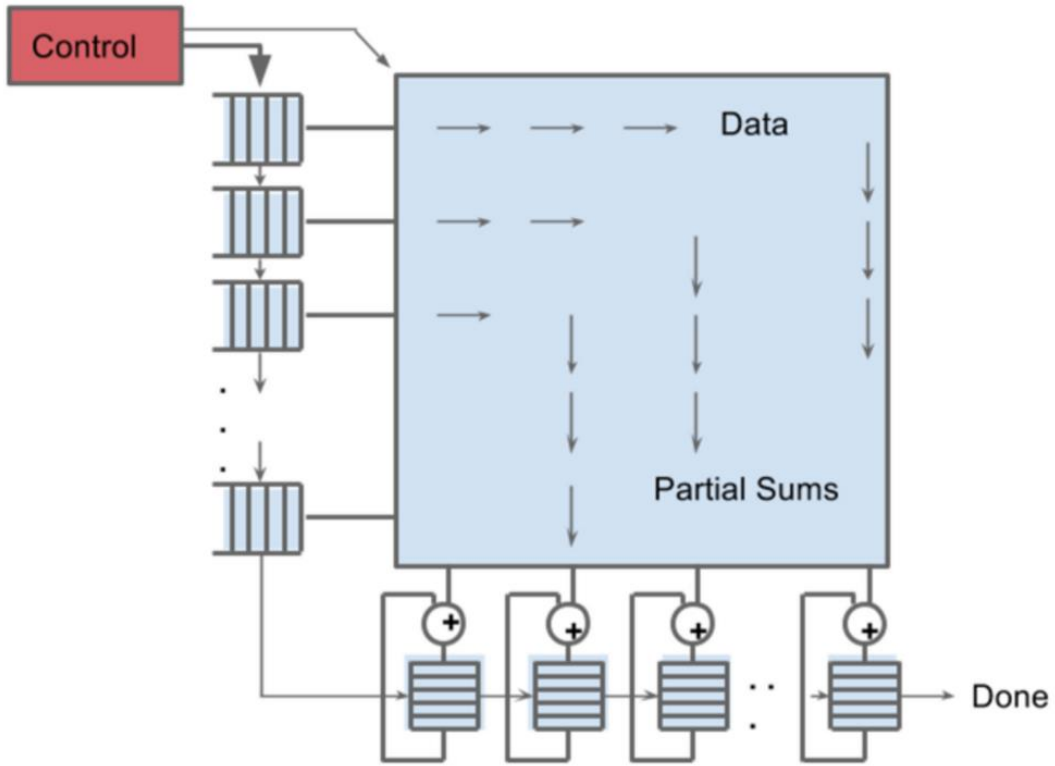
Example: Nvidia Turing (RTX 2080 TI FE / TU102)

Lanes = 4
M = 4
N = 4
Instances = 544

Total MACs = 34,816
Datatype = float16
Fmax GHz = 1.635
TFLOPS = 113.8

Source: https://www.nvidia.com/en-us/design-visualization/technologies/turing-architecture/

Example: TPU V1

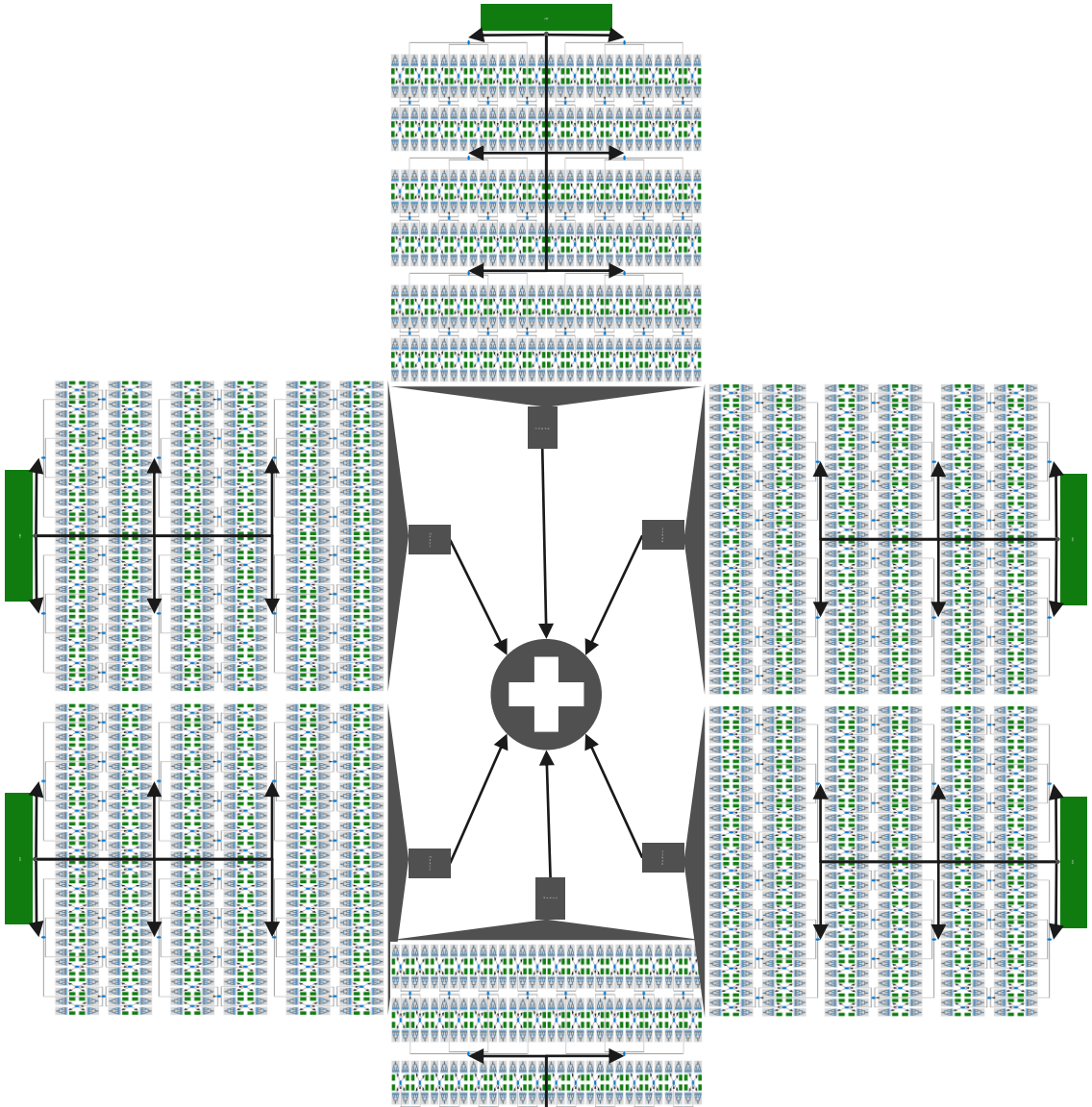


Lanes = 1
M = 256
N = 256
Instances = 1

Total MACs = 65,536
Datatype = int8
Fmax GHz = 0.7
TFLOPS/s = 91.8

Source: Jouppi et al., In-Datacenter Performance Analysis of a Tensor Processing Unit, ISCA'17

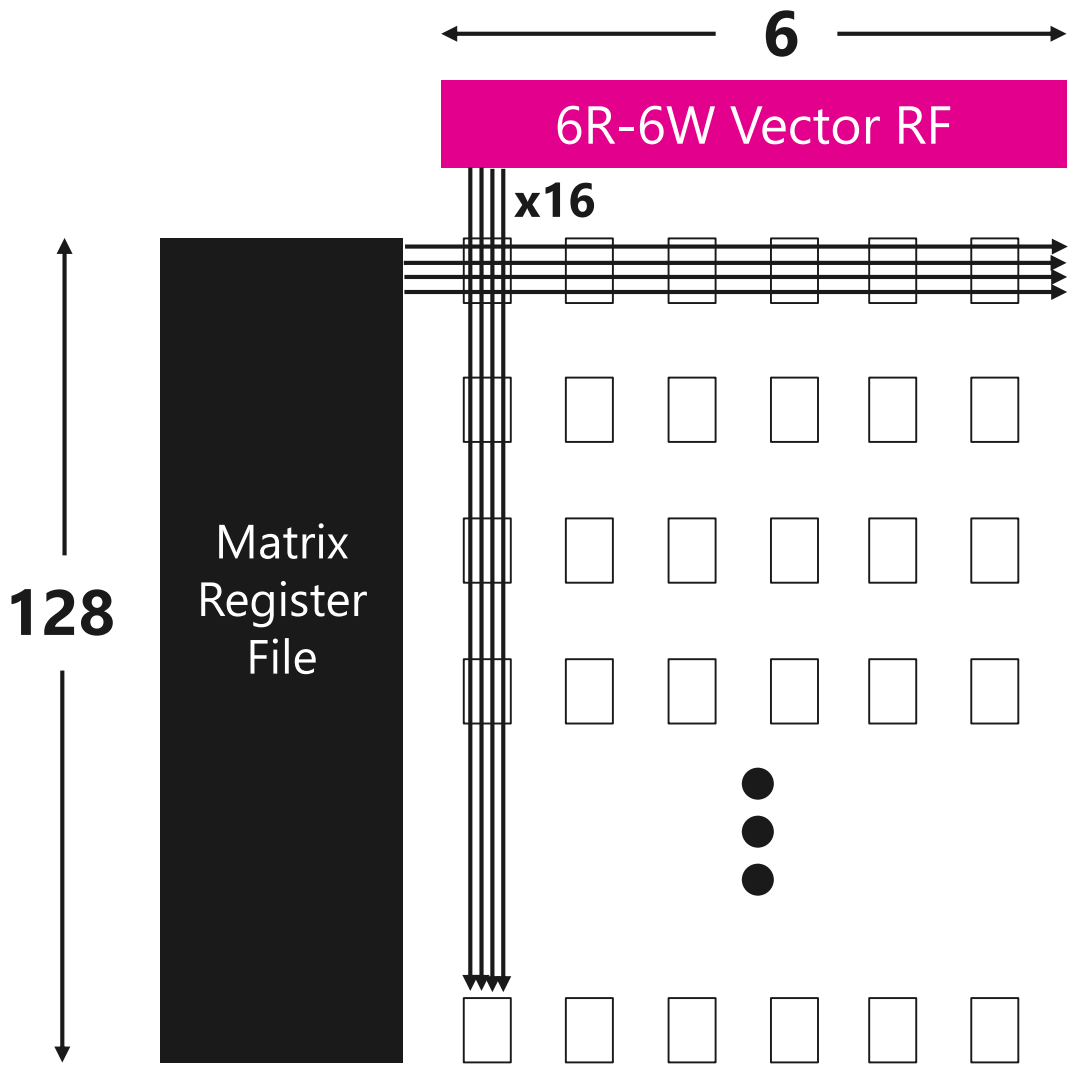
Example: Brainwave NPU (RNN-optimized/Stratix 10)



Lanes = 40
M = 400
N = 1
Instances = 6

Total MACs = 96,000
Datatype = msfp
Fmax GHz = 0.3
TFLOPS/s = 57.6

Example: Brainwave NPU (CNN-optimized/Arria10)



Lanes = 16
M = 128
N = 6
Instances = 1

Total MACs = 12,288
Datatype = msfp
Fmax GHz = 0.3
TFLOPS/s = 7.4

Brainwave NPU Instructions

Name	Description	IN	Operand 1	Operand 2	OUT
v_rd	Vector read	-	MemID	Memory index	V
v_wr	Vector write	V	MemID	Memory index	-
m_rd	Matrix read	-	MemID (NetQ or DRAM only)	Memory index	M
m_wr	Matrix write	M	MemID (MatrixRf or DRAM only)	Memory index	-
mv_mul	Matrix-vector multiply	V	MatrixRf index	-	V
vv_add	PWV addition	V	AddSubVrf index	-	V
vv_a_sub_b	PWV subtraction, IN is minuend	V	AddSubVrf index	-	V
vv_b_sub_a	PWV subtraction, IN is subtrahend	V	AddSubVrf index	-	V
vv_max	PWV max	V	AddSubVrf index	-	V
vv_mul	Hadamard product	V	MultiplyVrf index	-	V
v_relu	PWV ReLU	V	-	-	V
v_sigm	PWV sigmoid	V	-	-	V
v_tanh	PWV hyperbolic tangent	V	-	-	V
s_wr	Write scalar control register	-	Scalar reg index	Scalar value	-
end_chain	End instruction chain	-	-	-	-

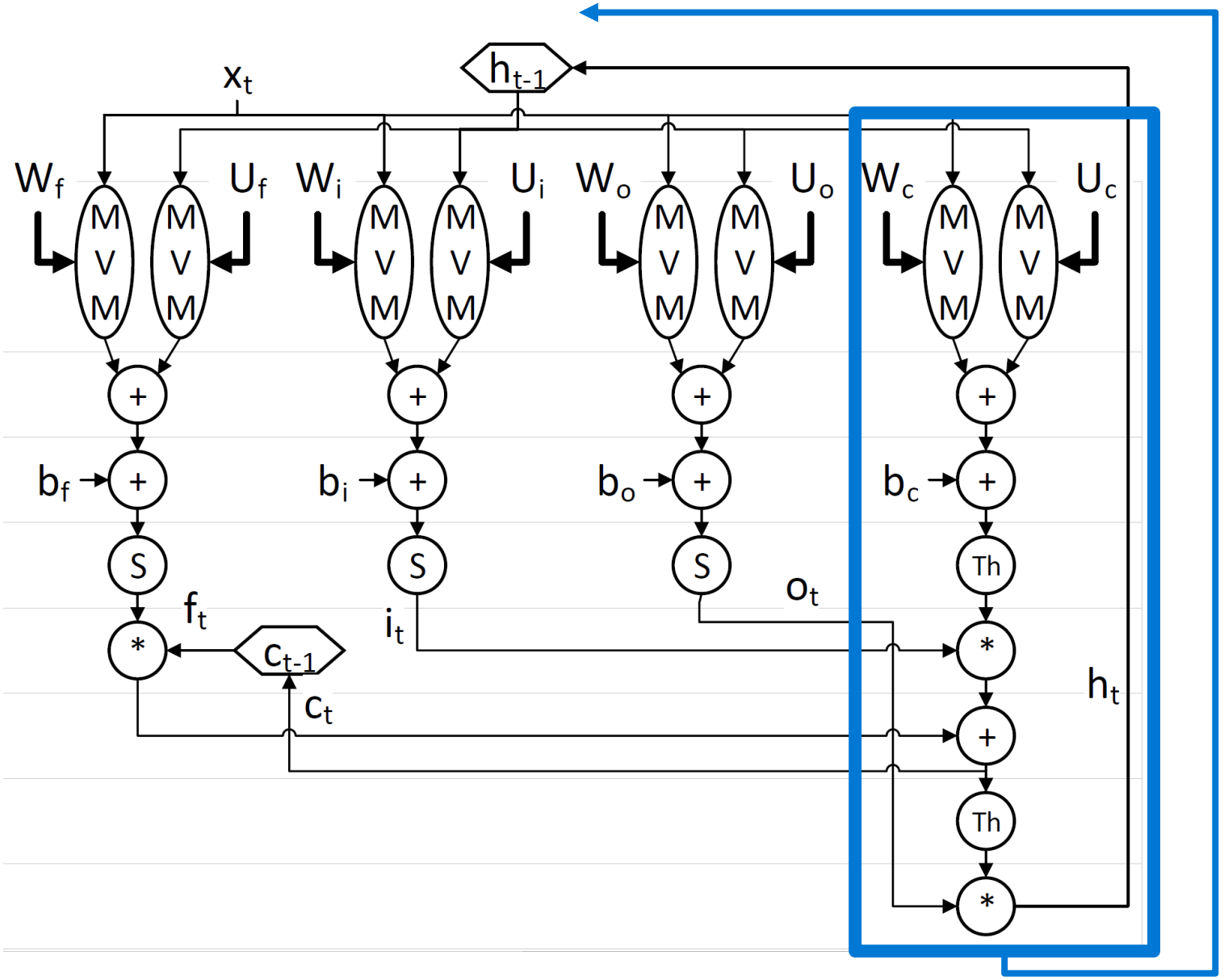
PWV = point-wise vector operation. IN = implicit input (V: vector, M: matrix, -: none). OUT = implicit output.

Example: LSTM

Model:
Long short-term memory (LSTM)

Applications:
NLP, speech, etc.

Type:
Recurrent neural network (RNN)



Example LSTM Program

```
1.  void LSTM(int steps) {
2.      for (int t = 0; t < steps; t++) {
3.          v_rd(NetQ);
4.          v_wr(InitialVrf, xt);
5.          v_rd(InitialVrf, xt);
6.          mv_mul(Wf);
7.          vv_add(bf);
8.          v_wr(AddSubVrf, xWf);
9.          v_rd(InitialVrf, h_prev);
10.         mv_mul(Uf);
11.         vv_add(xWf);
12.         v_sigm();
13.         vv_mul(c_prev);
14.         v_wr(AddSubVrf, ft_mod);
15.         v_rd(InitialVrf, h_prev);
16.         mv_mul(Uc);
17.         vv_add(xWc);
18.         v_tanh();
19.         vv_mul(it);
20.         vv_add(ft_mod);
21.         v_wr(MultiplyVrf, c_prev);
22.         v_wr(InitialVrf, ct);
23.         v_rd(InitialVrf, ct);
24.         v_tanh();
25.         vv_mul(ot);
26.         v_wr(InitialVrf, h_prev);
27.         v_wr(NetQ);
28.     }
29. }
```

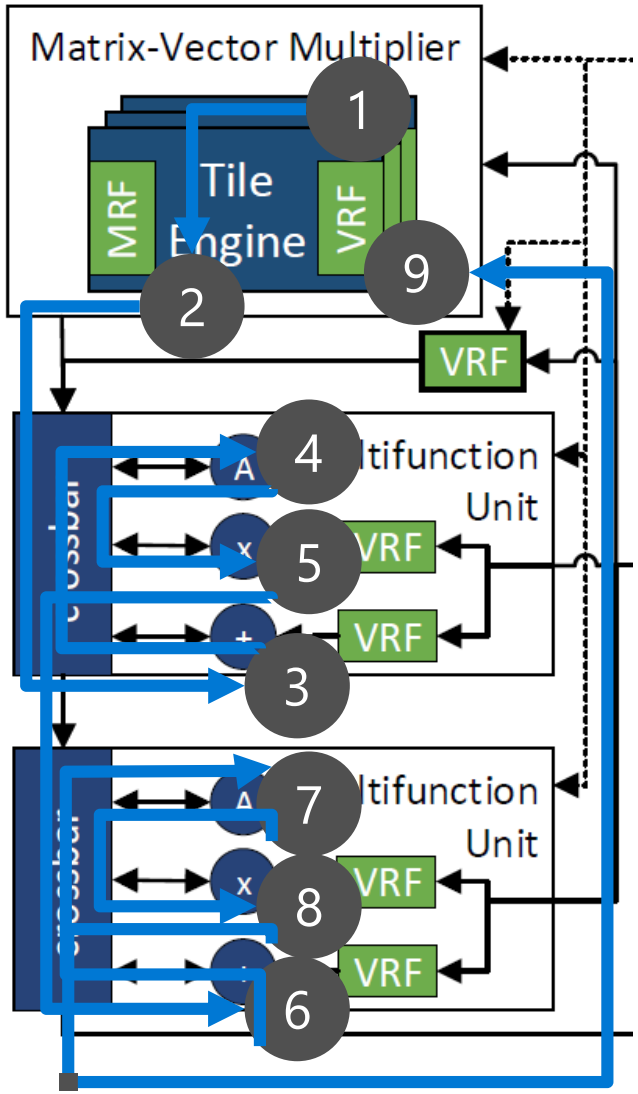
Microarchitecture

Executes a continuous stream of instruction chains

- Function units arranged to mirror chain structure
- Chaining reduces latency of critical path

Scaling M*V

- Memory bandwidth
- Tile engines
- Data types
- Scheduling

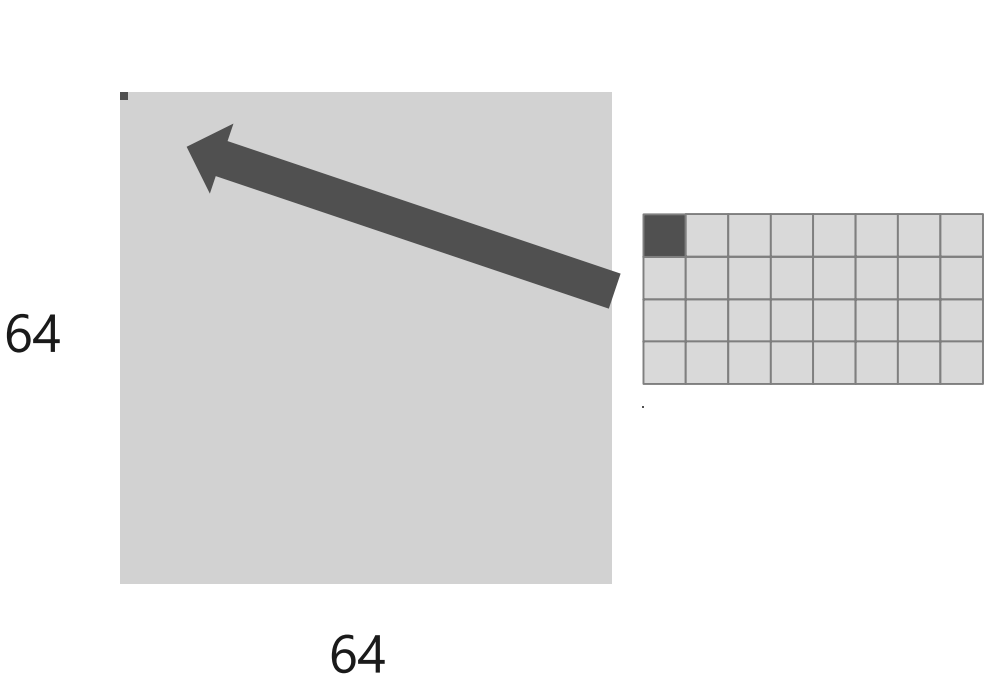


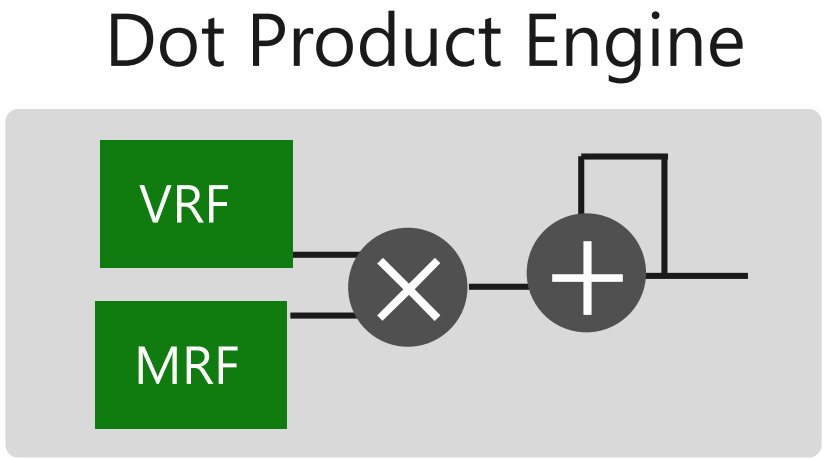
1. Read Vector x
2. M*V by W
3. Add Vector by b
4. Vector Tanh
5. Multiply Vector by i
6. Add Vector by f
7. Vector Tanh
8. Multiply by o
9. Write Vector h

Scaling M*V: Single Spatial Unit

Start with a primitive 1 MAC for M*V

Vector and matrix register files (VRF, MRF) read 1 word/cycle



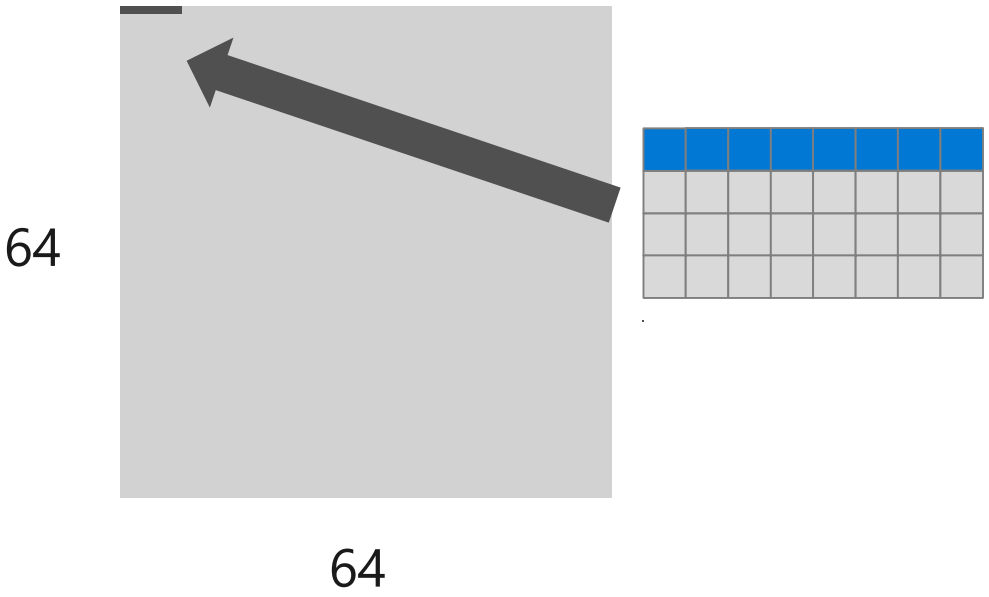


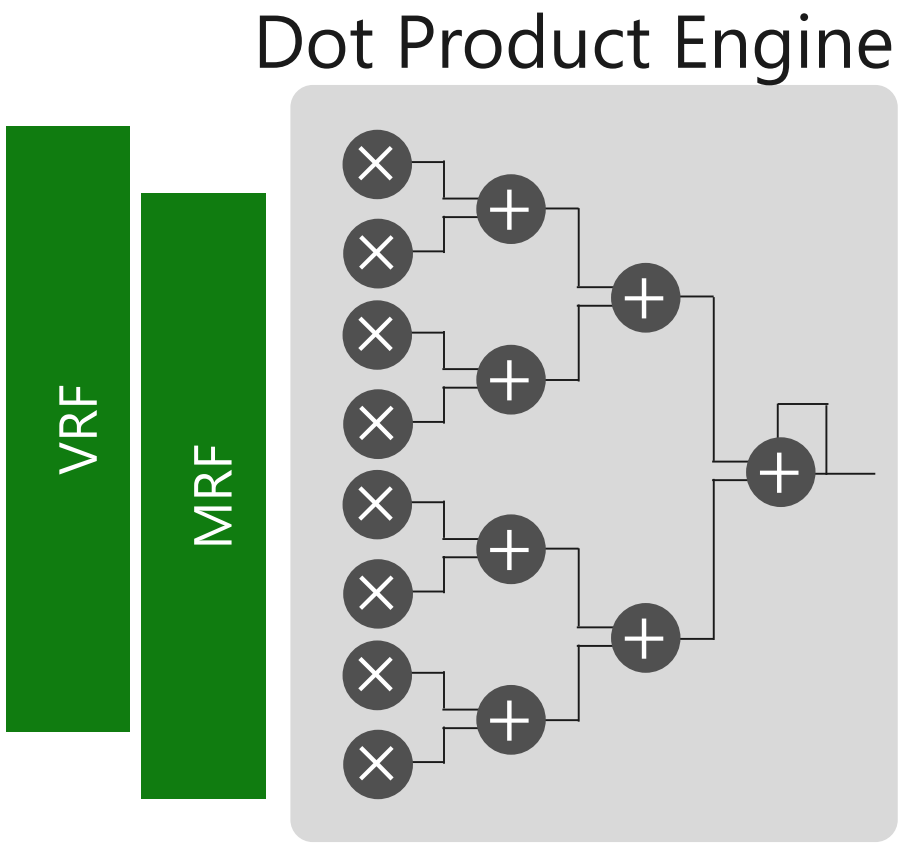
Scaling M*V: Multi-Lane Vector Spatial Unit

Compute lanes x8 = 8 MACs

Column parallelism

Scaling limit: more lanes = bigger RFs





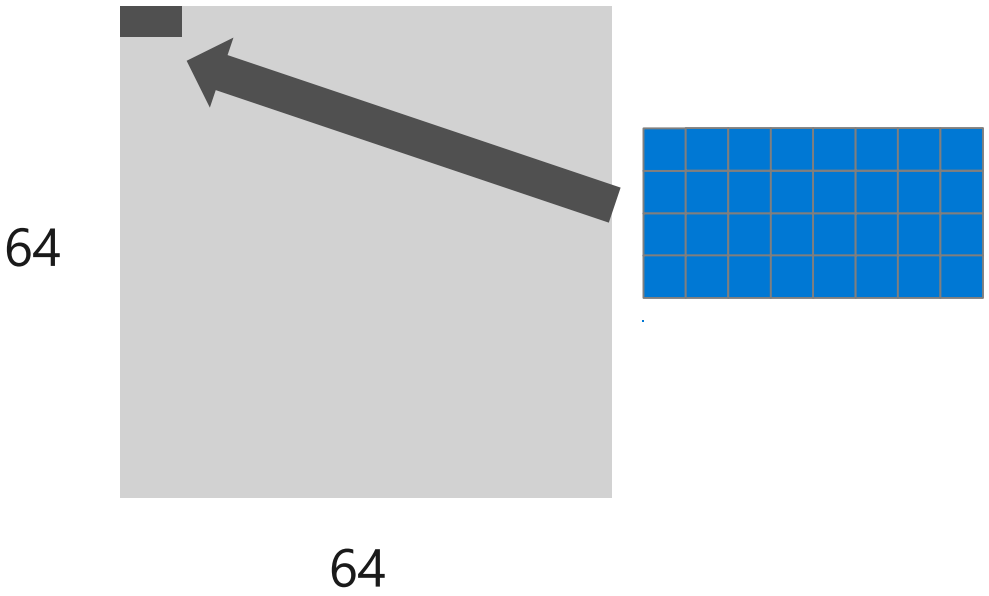
Scaling M*V: Vector Spatial Unit Replication

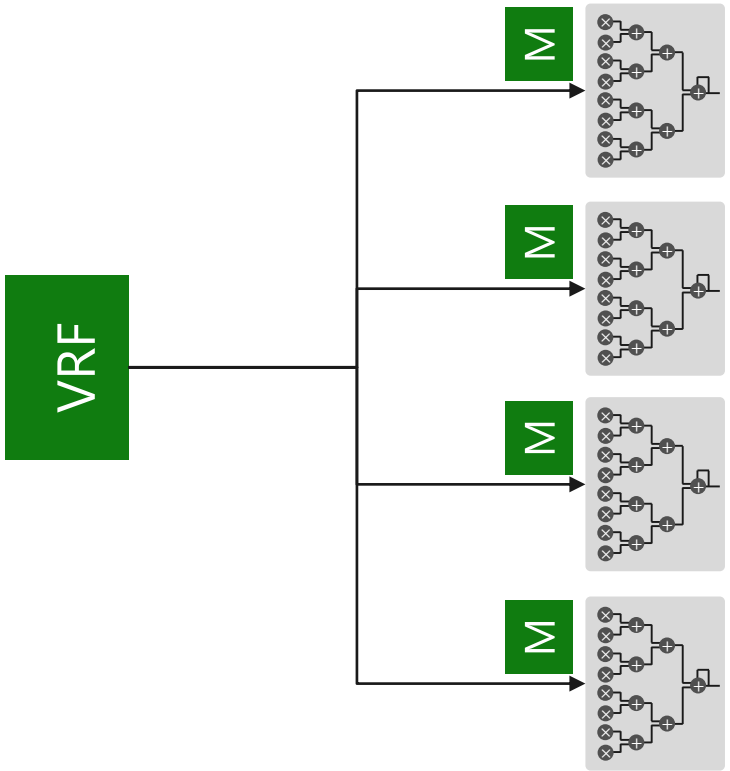
Replicate DPE x4 = 32 MACs

Matrix row parallelism, distributed MRF

Reuse VRF data across dot products

Scaling limit: rows = DPEs

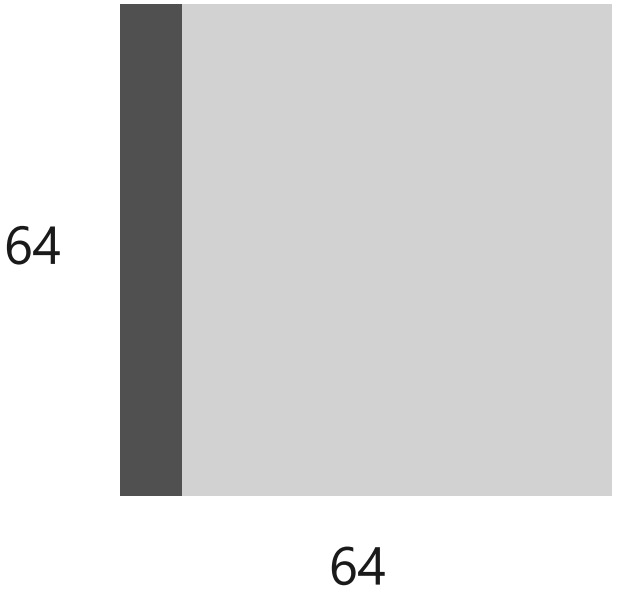


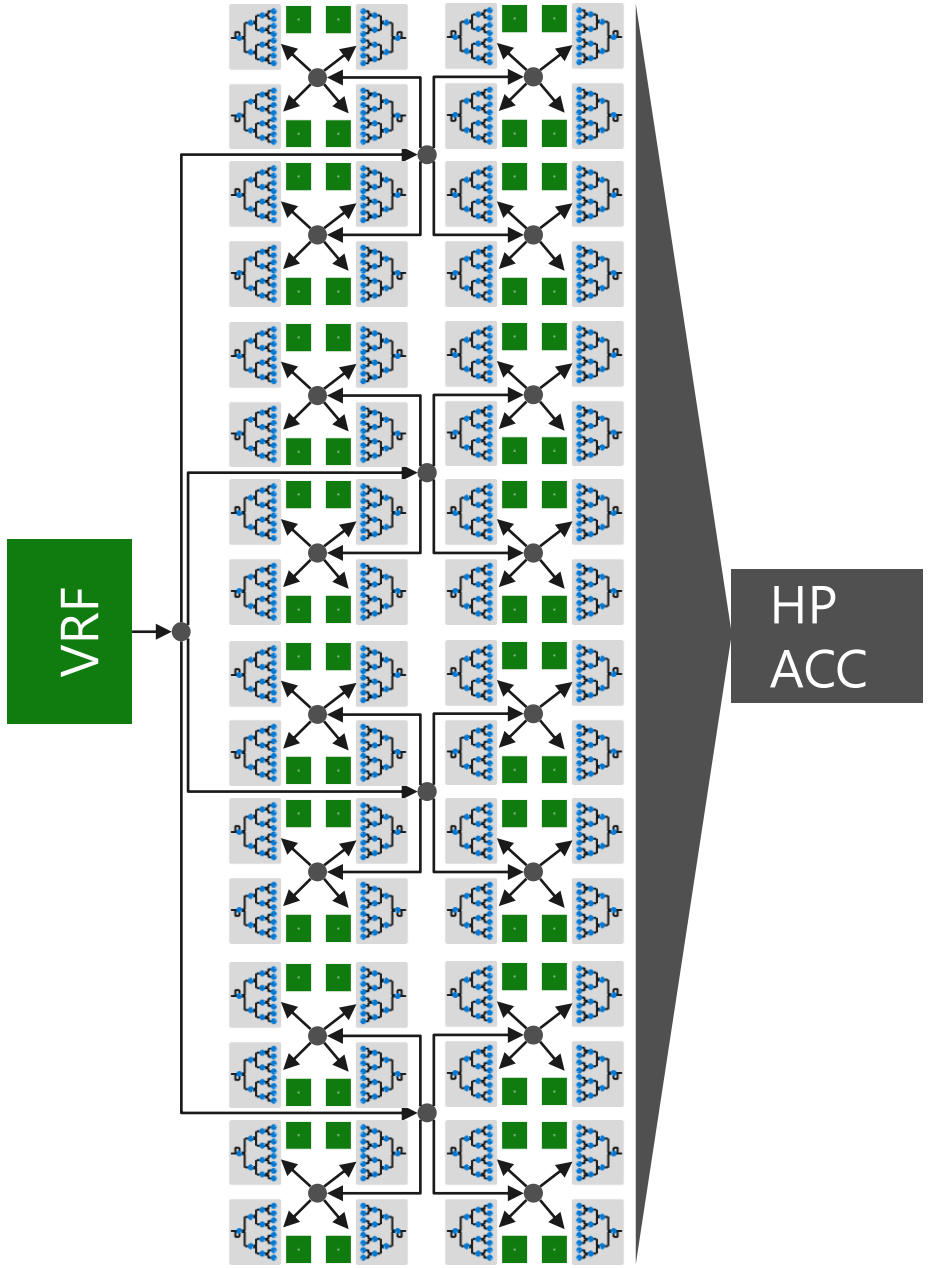


Scaling M*V: Scalable Replication

Tile engine: native size M*V tile

- High precision accumulator (HP ACC)
- Registered input fan-out tree
- Result fan-in tree (muxs)



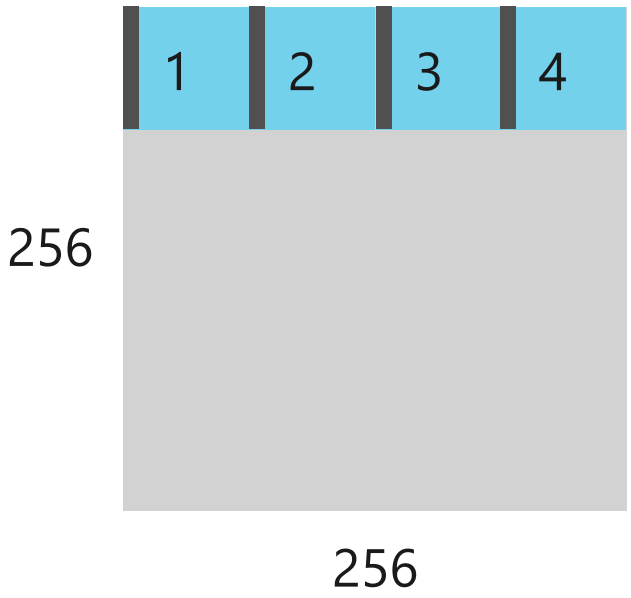


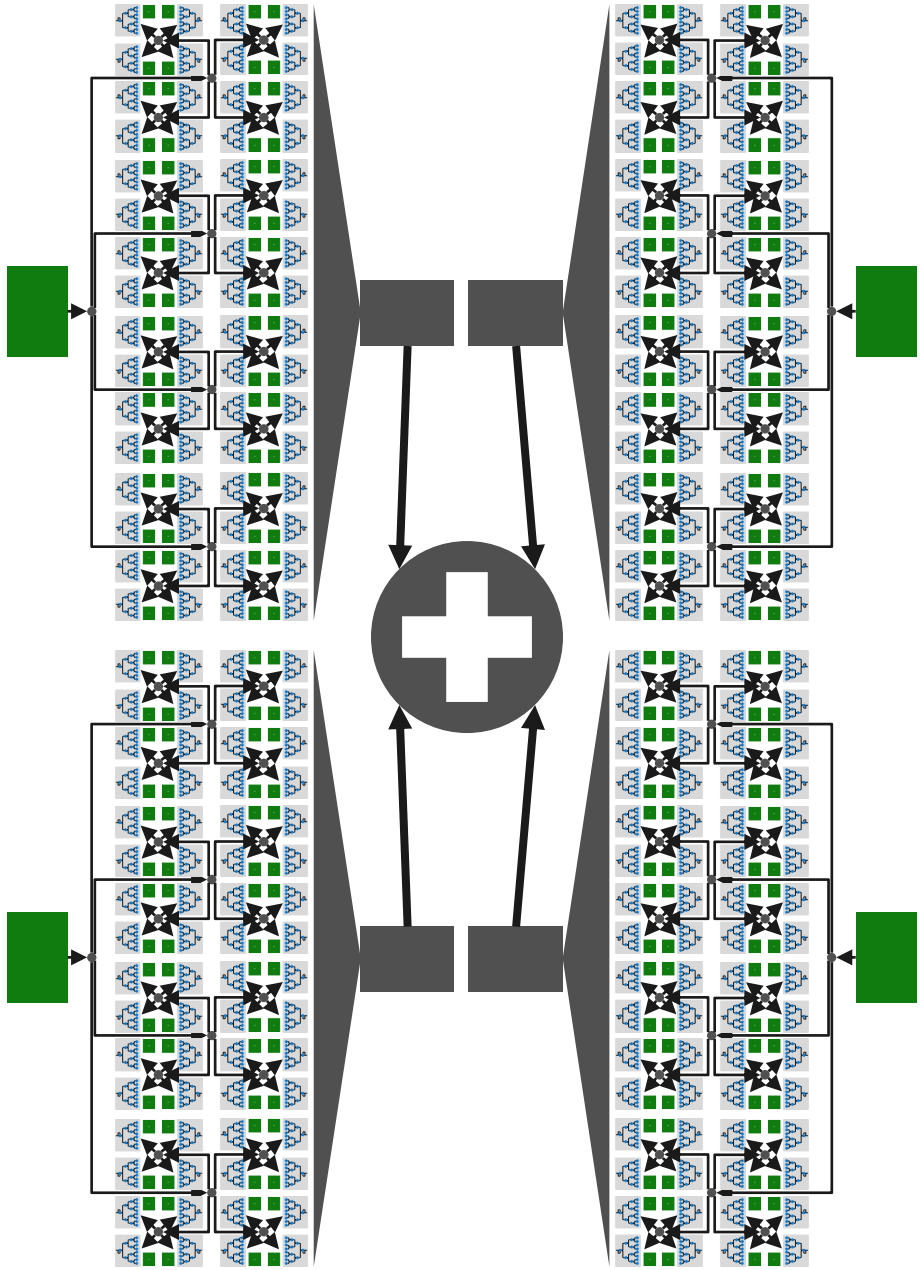
Scaling M*V: Tiling

Large matrices: tile parallelism

Add-reduction unit

Scaling limit: area, matrix columns





Scaling M*V: Narrow Precision Data Types

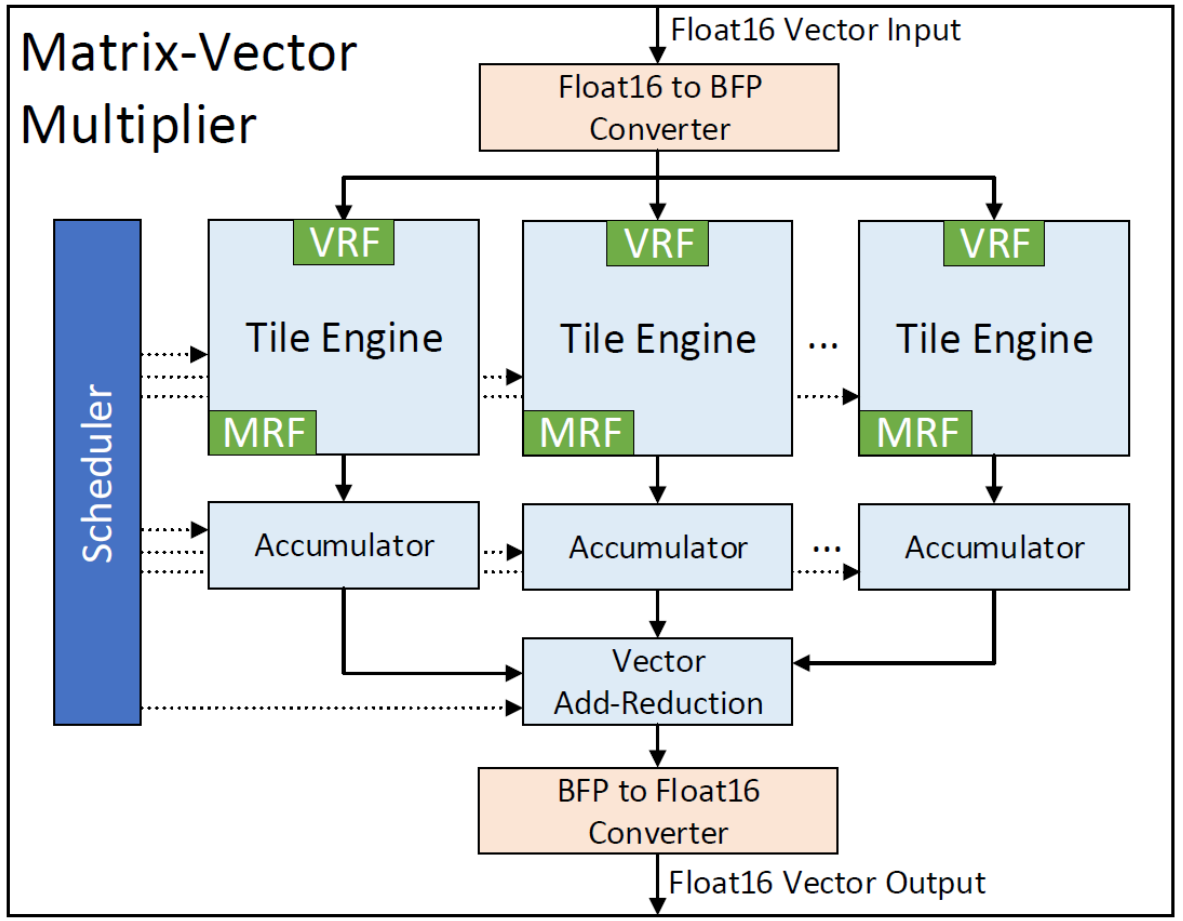
FP8 - FP11 sufficient

e.g., 1-bit sign, 5-bit exponent, 2-bit mantissa

Block floating point

Shared exponent for native vectors

Integer arithmetic in tile engines

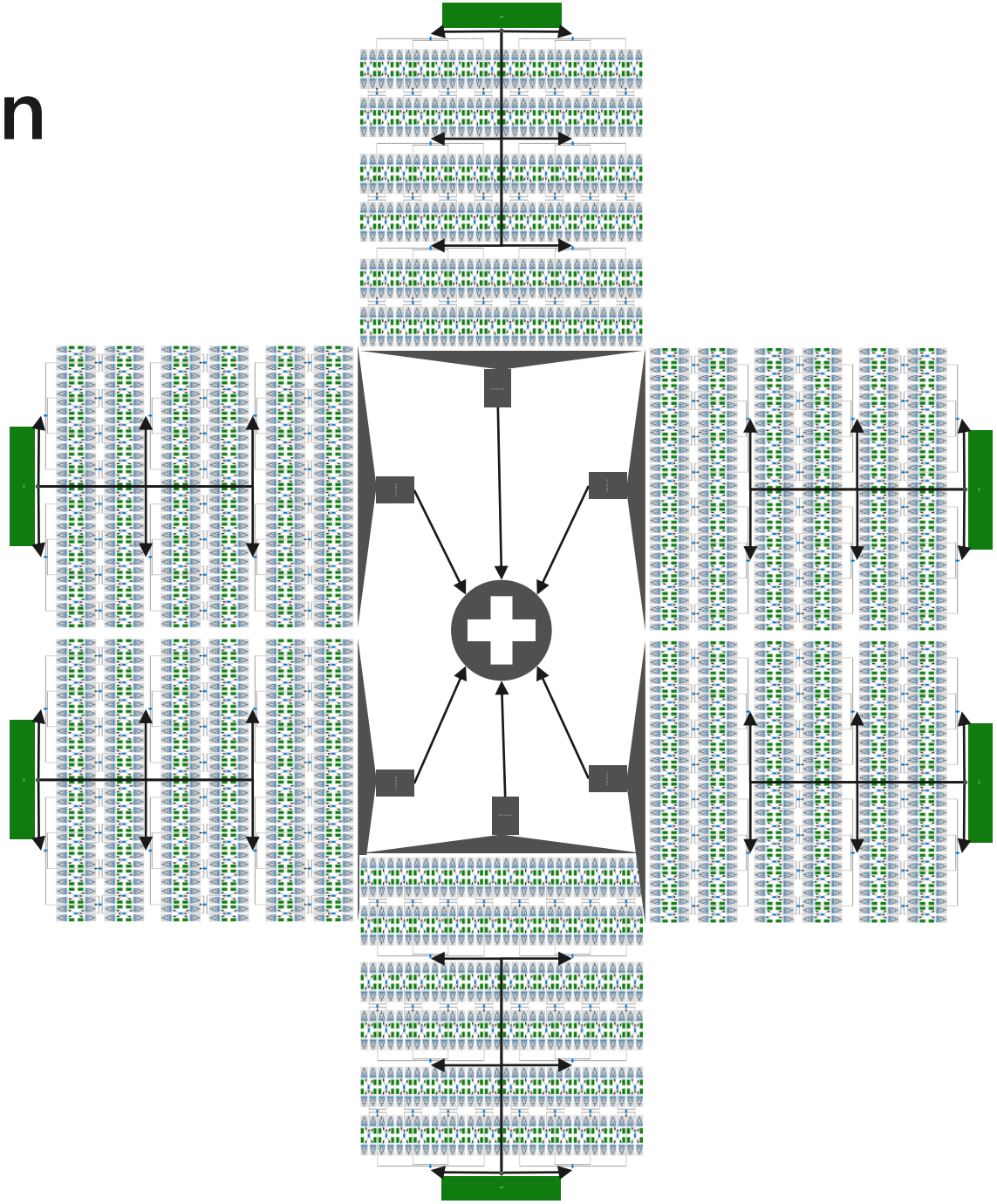


Scaling M*V: Narrow Precision

Fully scaled Block FP8 on Stratix 10

- 6 Tile Engines
- 400 dot products each
- 40 lanes each
- **Total =** 6 * 400 * 40 = **96,000 MACs**

MRF is 96,000 words/cycle



NPU Evaluation

BW NPU Instance	#MV Tiles	#Lanes	Native Dim.	MRF Size	#MFUs	Target Device	#ALMs (%)	#M20Ks (%)	DSPs (%)	Freq. (MHz)	Peak TFLOPS
BW_S5	6	10	100	306	2	Stratix V D5	149641 (87%)	1192 (59%)	1047 (66%)	200	2.4
BW_A10	8	16	128	512	2	Arria 10 1150	216602 (51%)	2171 (80%)	1518 (100%)	300	9.8
BW_S10	6	40	400	306	2	Stratix 10 280	845719 (91%)	8192 (69%)	5245 (91%)	250	48

	Titan Xp	BW_S10
Numerical Type	Float32	BFP (1s.5e.2m)
Peak TFLOPS	12.1	48.0
TDP (W)	250	125
Process	TSMC 16nm	Intel 14nm

TABLE IV
EXPERIMENT HARDWARE SPECIFICATIONS

Benchmark	Device	Latency (ms)	TFLOPS	% Utilization
GRU h=2816 t=750	SDM	1.581	-	-
	BW	1.987	35.92	74.8
	Titan Xp	178.60	0.40	3.3
GRU h=2560 t=375	SDM	0.661	-	-
	BW	0.993	29.69	61.8
	Titan Xp	74.62	0.40	3.3
GRU h=2048 t=375	SDM	0.438	-	-
	BW	0.954	19.79	41.2
	Titan Xp	51.59	0.37	3.0
GRU h=1536 t=375	SDM	0.266	-	-
	BW	0.951	11.17	23.3
	Titan Xp	31.73	0.33	2.8
GRU h=1024 t=1500	SDM	0.558	-	-
	BW	3.792	4.98	10.4
	Titan Xp	59.51	0.32	2.6
GRU h=512 t=1	SDM	0.00017	-	-
	BW	0.013	0.25	0.5
	Titan Xp	0.06	0.05	0.4

NPU Evaluation

Stratix V RNN-optimized NPU for Bing Production

Bing TP1			
	CPU-only	**Brainwave-accelerated**	**Improvement**
Model details	GRU 128x200 (x2) + W2Vec	LSTM 500x200 (x8) + W2Vec	Brainwave-accelerated model is > 10X larger and > 10X lower latency
End-to-end latency per Batch 1 request at 95%	9 ms	0.850 ms	

Arria 10 CNN-optimized NPU

	Nvidia P40	BW_CNN_A10
Technology node	16nm TSMC	20nm TSMC
Framework	TF 1.5 + TensorRT 4	TF + BW
Precision	INT8	BFP (1s.5e.5m)
IPS (batch 1)	461	559
Latency (batch 1)	2.17 ms	1.8 ms

Lessons

- It is challenging to move the needle at system scale
  - Solving the “right” problem is important – what is your E2E workload, where is time spent
  - Peak FLOPS alone is a poor metric, an efficient architecture and E2E perf more important
- Cloud verticalization = unfair advantage
  - Creates opportunity to co-optimize algorithms and hardware (e.g., pinning and quantization)
- The value of agility
  - Using FPGAs at cloud scale enabled us to iterate rapidly with optionality to harden
  - Software teams move faster when the hardware is available right away

Project Brainwave
for the Masses

Azure Machine Learning and Project Brainwave

End-to-end data science platform

Intelligent data preparation, model training, and operationalization to cloud and edge

Model management, telemetry, A/B testing, etc.

Enterprise grade: Compliant and secure

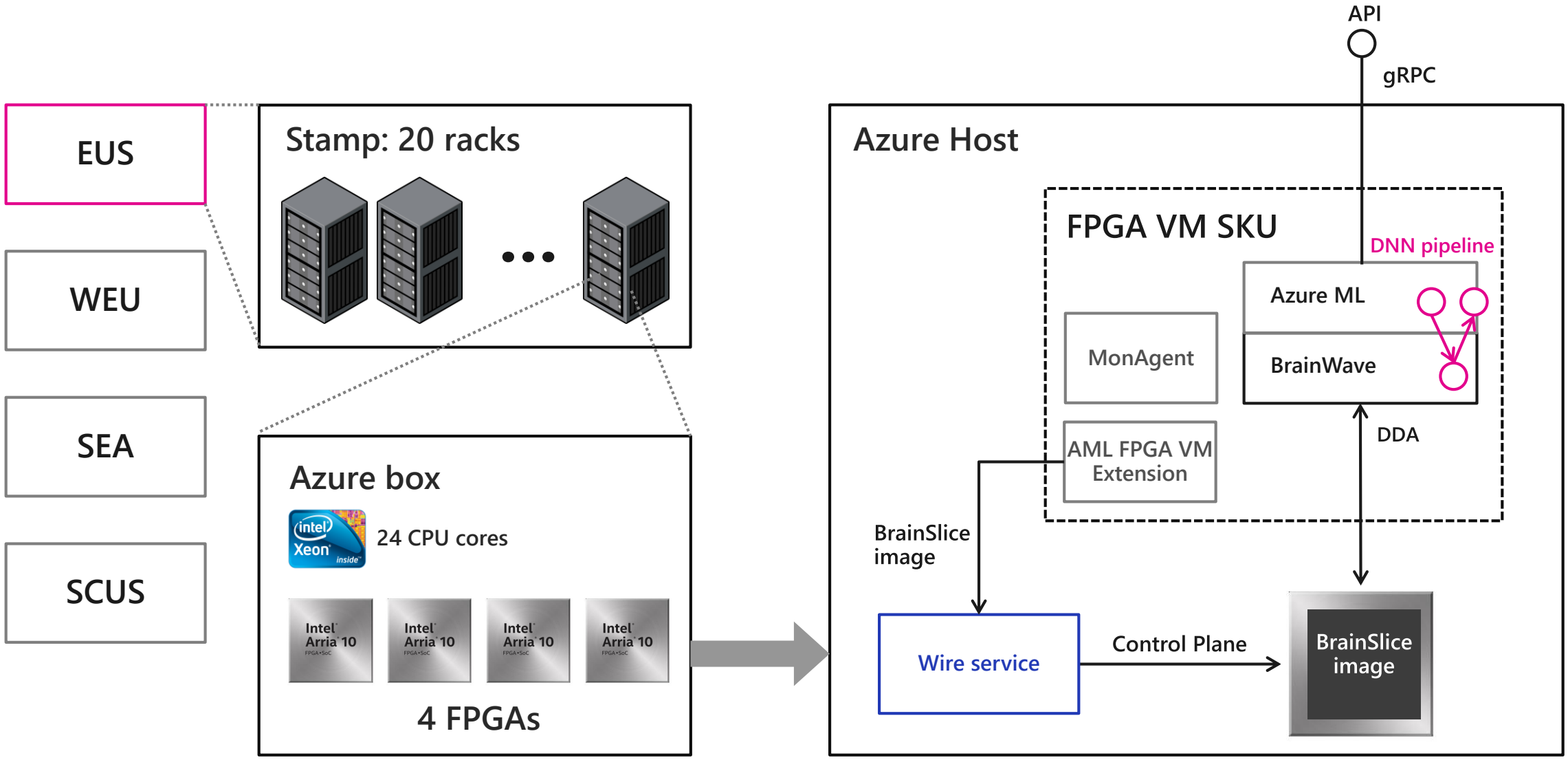
Code in Python

Use Python and TensorFlow to create models to deploy to FPGAs

Serverless architecture

Deploy a model and get a gRPC API for inferencing

FPGA service infra and stack



Deploying a model

Azure ML

Service def

Python and TensorFlow

Load AML SDK

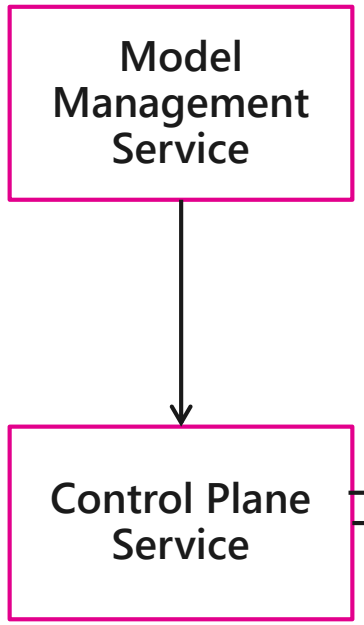
Transform pictures into tensors

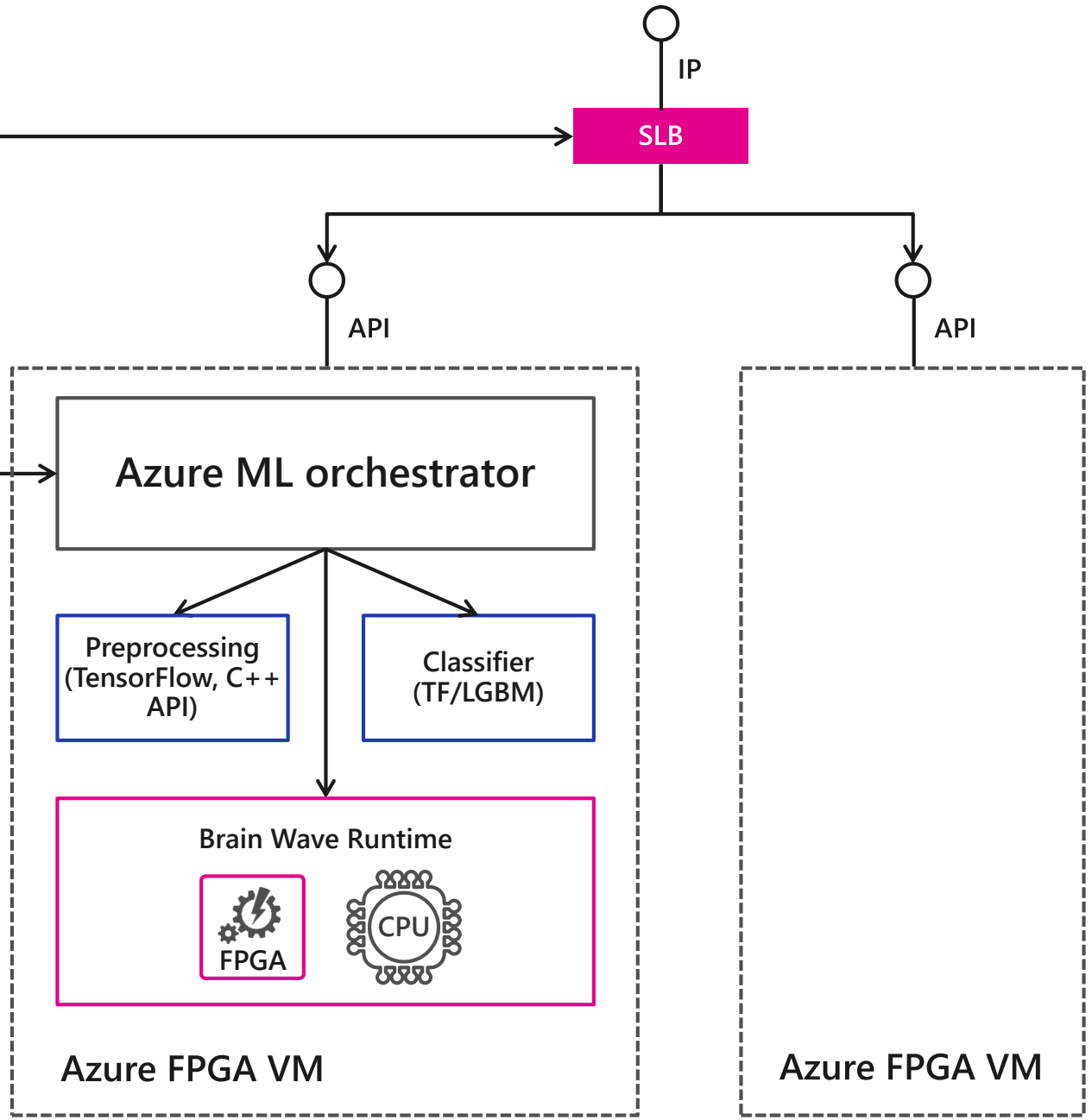
Load pre-trained quantized RN50 without last layer as deep featurizer

Add classifier or additional layers for classification

Featurize images and train classifier

Local, Spark CPU cluster, or GPU cluster





Azure IoT edge

Cloud services at the edge

Azure ML, Azure Stream Analytics, Azure Functions, custom

Manage from the cloud

Devices and services from Azure Portal

Flexible connectivity

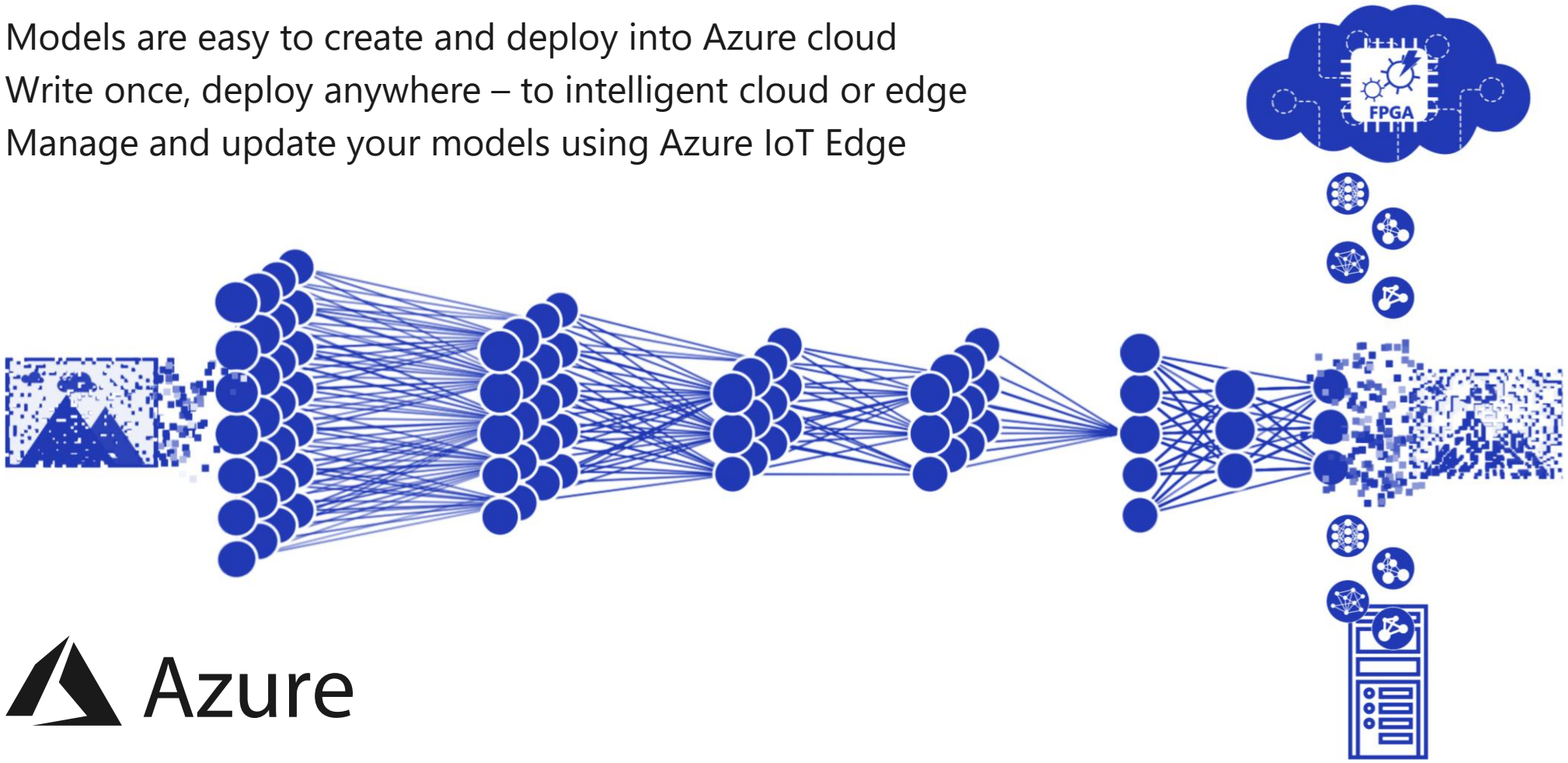
Intermittent, low, or no connectivity

Reduced latency and cost

Bring compute to the data, reduced bandwidth cost

Try Azure ML with Project Brainwave on cloud or edge

Models are easy to create and deploy into Azure cloud
Write once, deploy anywhere – to intelligent cloud or edge
Manage and update your models using Azure IoT Edge

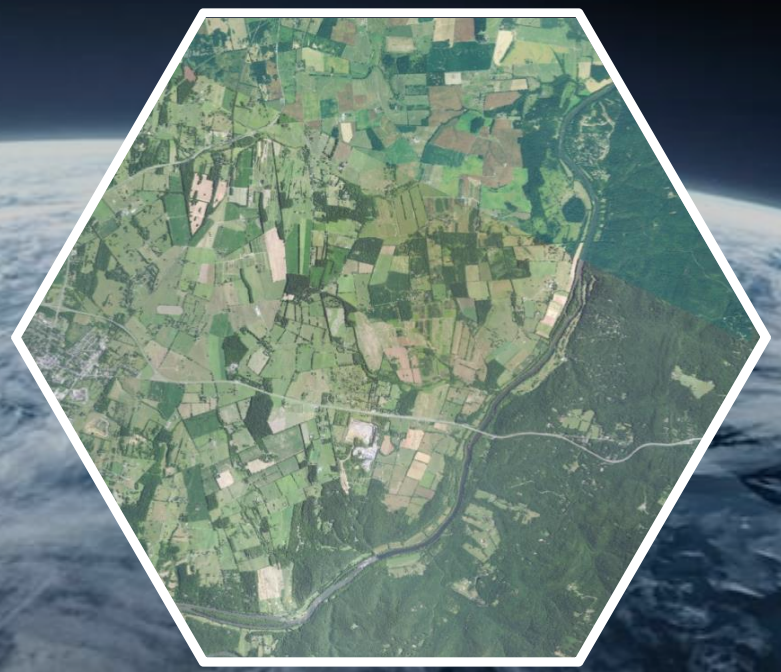


Azure

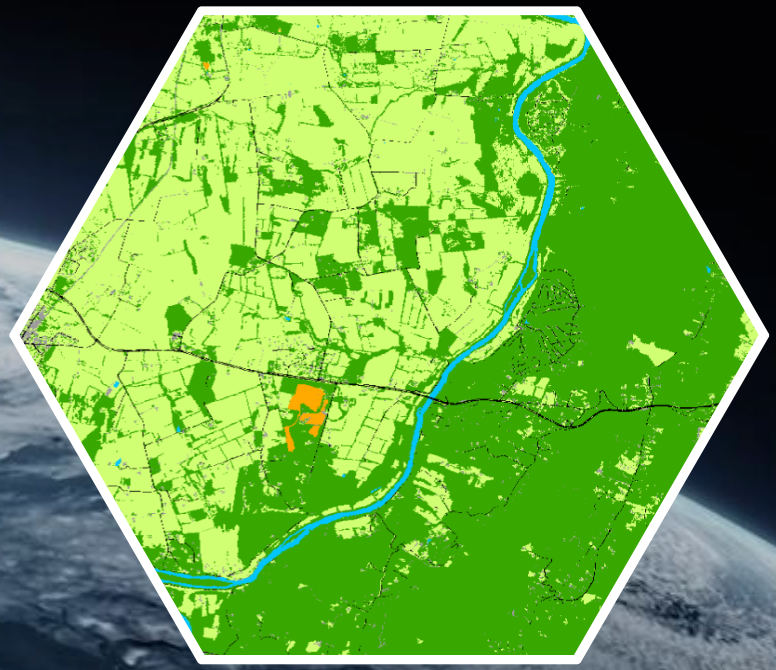
http://aka.ms/rtai
brainwave-edge@microsoft.com

Planet scale Real-Time Inferencing for GeoAI

Freely Available Imagery



Labeled Training Data



Inferred Land Cover Map

Using Project Brainwave for Land Use Mapping

FPGAs for ultra-fast inferencing different types of land use for the entire United States (ESRI NAIP Data, 20+ TB)

Data

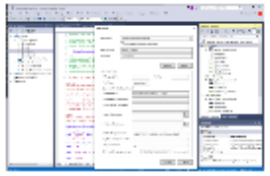
esri Satellite Images for the Entire US

NAIP Data
20TB, 200M images

Stored on **Azure Premium Storage**

Build

Geo AI Data Science Virtual Machine



Visual Studio Tools for AI

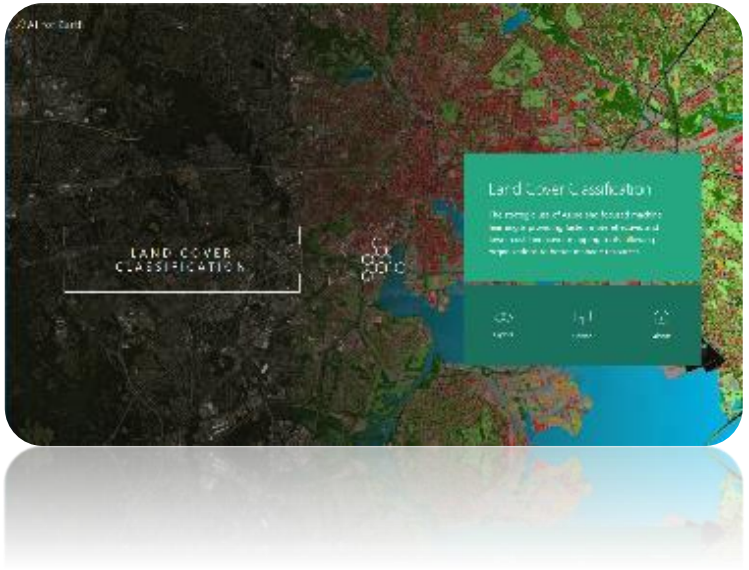
Train

Azure Machine Learning

Azure Batch AI

Deploy

Intelligent Apps



Land Classification Model
ResNet-50

Ultra-fast Inferencing using FPGAs

DATA ---------- INTELLIGENCE ---------- ACTION

Real-Time Low-Latency Inferencing with FPGAs

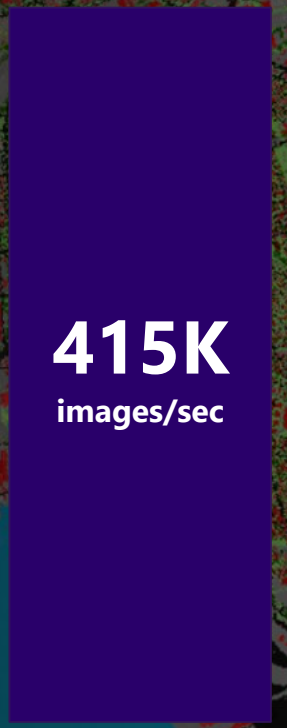
Setup:

- 800 FPGAs on Azure
- 195 Million Images; 20TB
- Real-time inferencing 1 image at a time

Results:

- 415K inferences/second @ 1.8ms latency
- 10.6 minutes total
- Order of magnitude better Price/Perf over CPU & GPU (V100 with TensorRT)

Cost: $42



Conclusions

Project Brainwave enables low-cost, real-time inferencing at cloud scale

- Codesigned across algorithm, system, and hardware
- Enabled with flexible, real-time NPU architecture
- Currently deployed in scale 1P services such as Bing
- Available to 3P customers through AzureML (cloud and edge)
- More models and customizability coming soon

Thank you!

Contact

Eric Chung (erchung@microsoft.com)

Papers

A Configurable Cloud-Scale DNN Processor for Real-Time AI, ISCA'18

Serving DNNs in Real Time at Datacenter Scale with Project Brainwave, IEEE MICRO Hotchips

A Cloud-Scale Acceleration Architecture, MICRO'16

A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services, ISCA'14

Links

https://www.microsoft.com/en-us/research/project/project-catapult/

https://aka.ms/aml-real-time-ai