Deep Learning Acceleration via Low Precision Computing

Zhaoxia (Summer) Deng
AI System Co-design @ facebook

Team Introduction

- AI System Co-design team mission:
  - AI application-driven sw & hw co-design through
    - High performance numerical and architectural optimizations
    - HW performance modeling and simulations
- Expertise
  - HPC and parallel algorithms
  - Computer architecture
  - Performance optimization and modeling
  - Numerical linear algebra, ML, and graph analytics

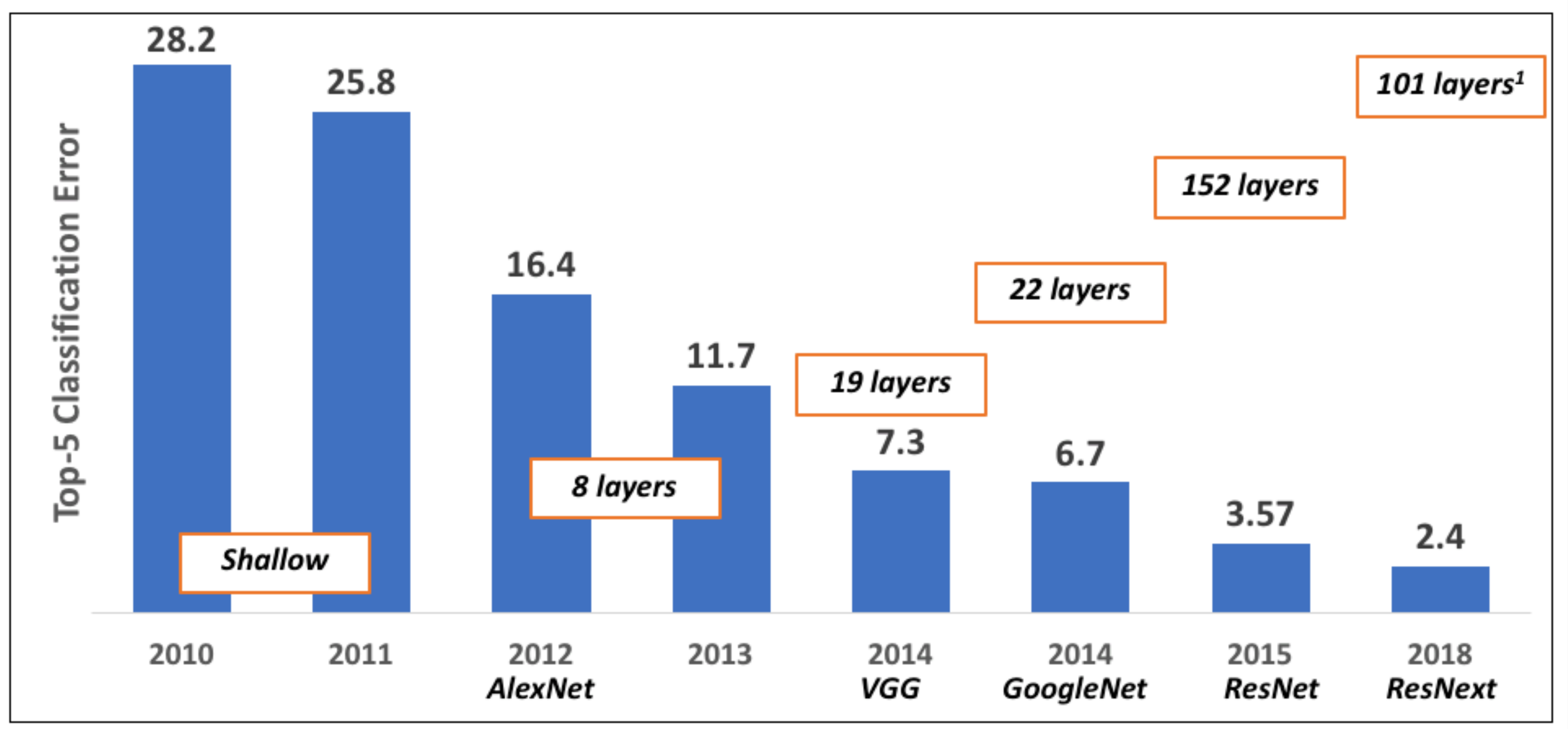
Agenda

- Facebook AI workload characteristics
- Low precision computing
  - Reduced precision floating point optimization
  - Fixed point quantization
- AI system co-design for low precision computing
  - Model co-design
  - Hardware co-design

Agenda

- **Facebook AI workload characteristics**
- Low precision computing
  - Reduced precision floating point optimization
  - Fixed point quantization
- AI system co-design for low precision computing
  - Model co-design
  - Hardware co-design

AI Growth and Its Drivers



Big and better data

Better algorithms

More compute

AI Driven Services at Facebook

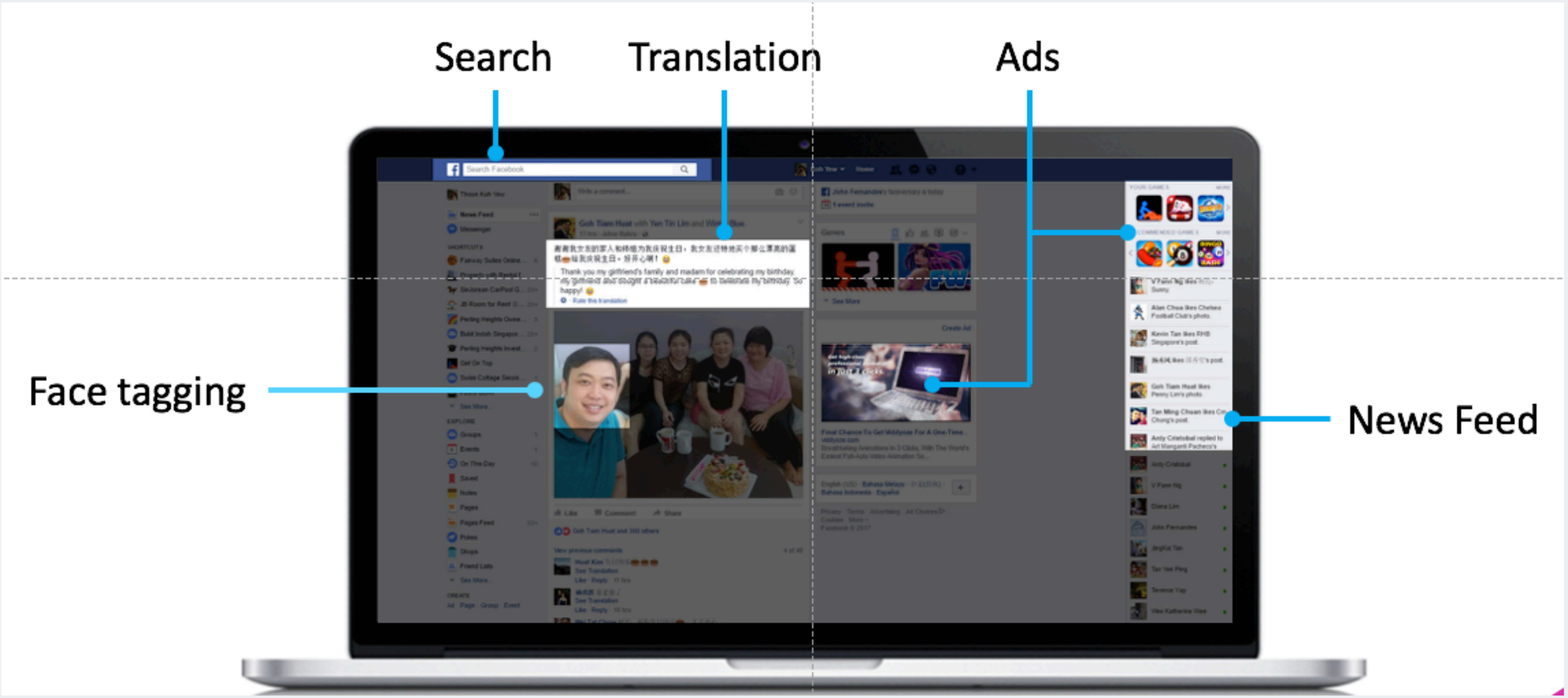
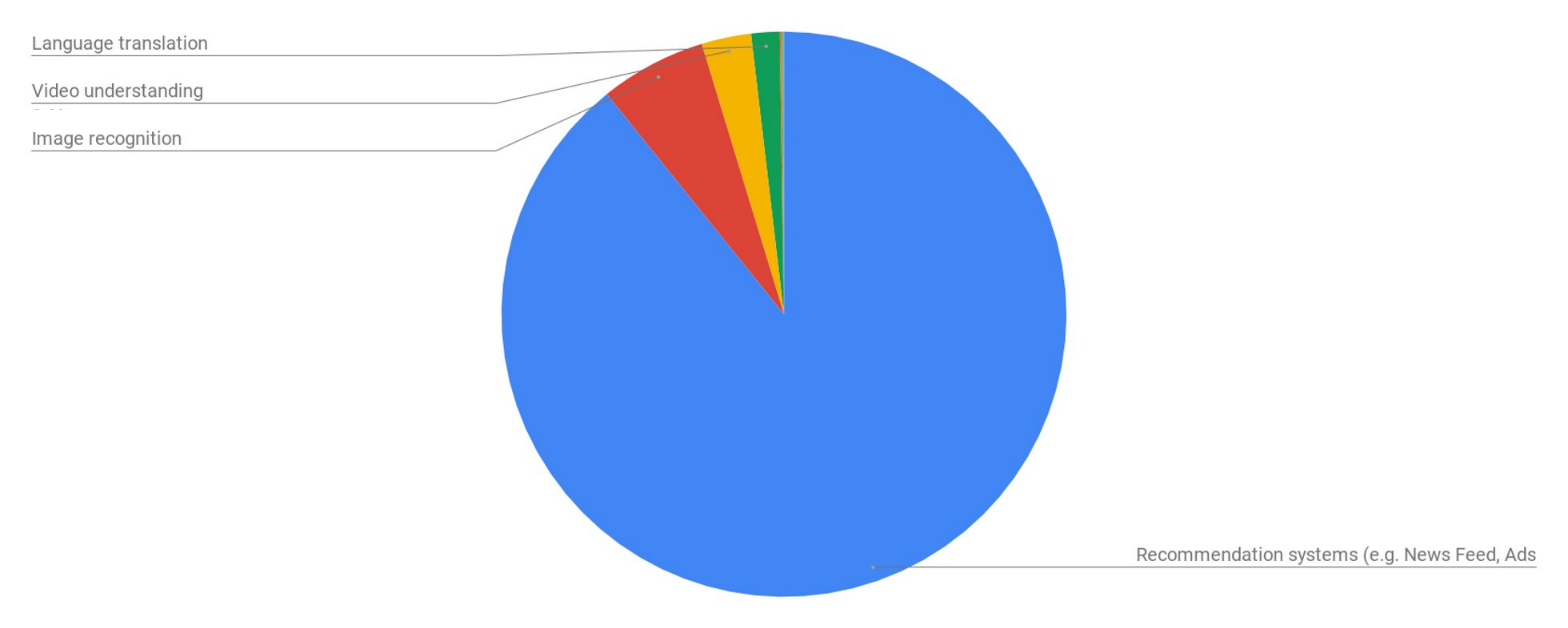


Figure credit: Misha Smelyanski

AI Execution Flow

AI Inference in Facebook Datacenters



Workload characteristics

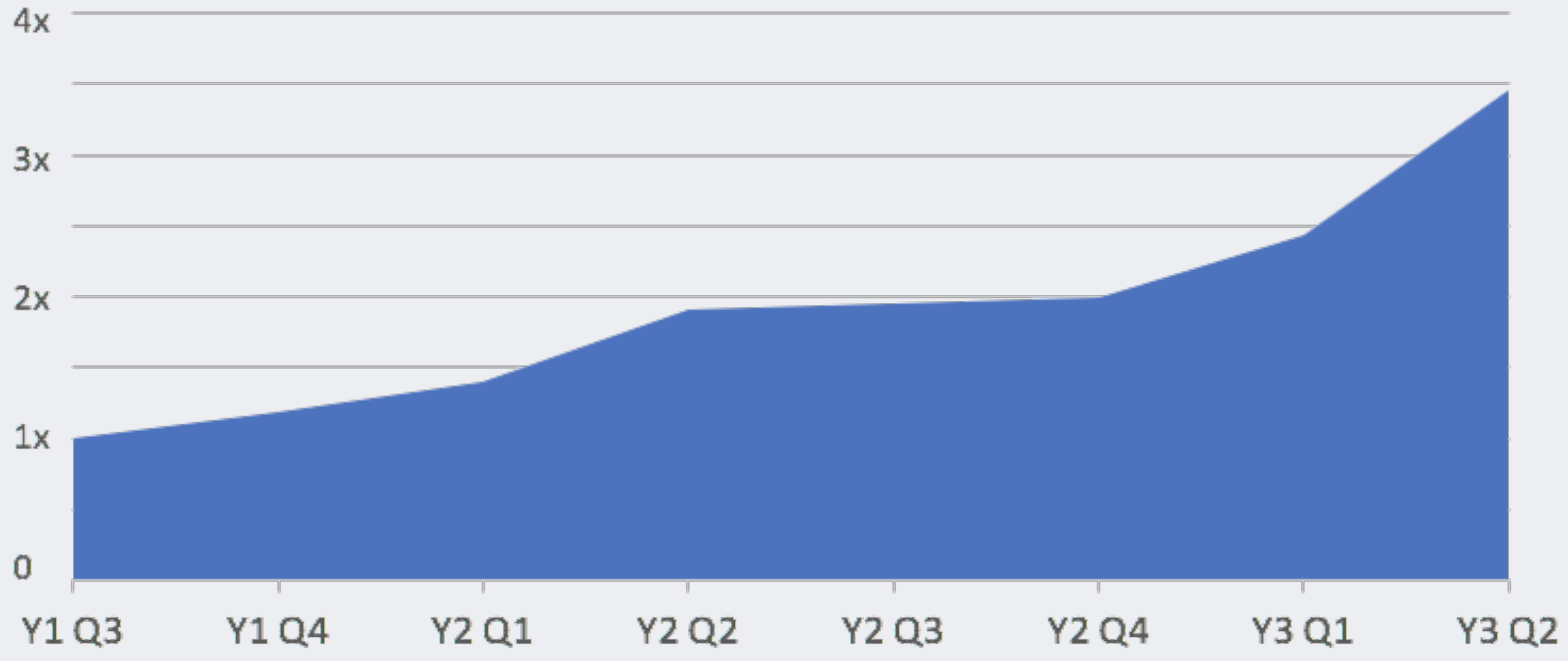
Category	Model Types	Model Size (# params)	Max. Live Activations	Op. Intensity (w.r.t. weights)	Op. Intensity (w.r.t. act & weights)
Recommendation	FCs	1-10M	> 10K	20-200	20-200
	Embeddings	>10 Billion	> 10K	1-2	1-2
Computer Vision	ResNeXt101-32x4-48	43-829M	2-29M	avg. 380 Min. 100	Avg. 188 Min. 28
	Faster-RCNN (with ShuffleNet)	6M	13M	Avg. 3.5K Min. 2.5K	Avg. 145 Min. 4
	ResNeXt3D-101	21M	58M	Avg. 22K Min. 2K	Avg. 172 Min. 6
Language	seq2seq	100M-1B	>100K	2-20	2-20

Deep Learning Inference in Facebook Data Centers:
Characterization, Performance Optimizations and Hardware Implications
https://arxiv.org/abs/1811.09886

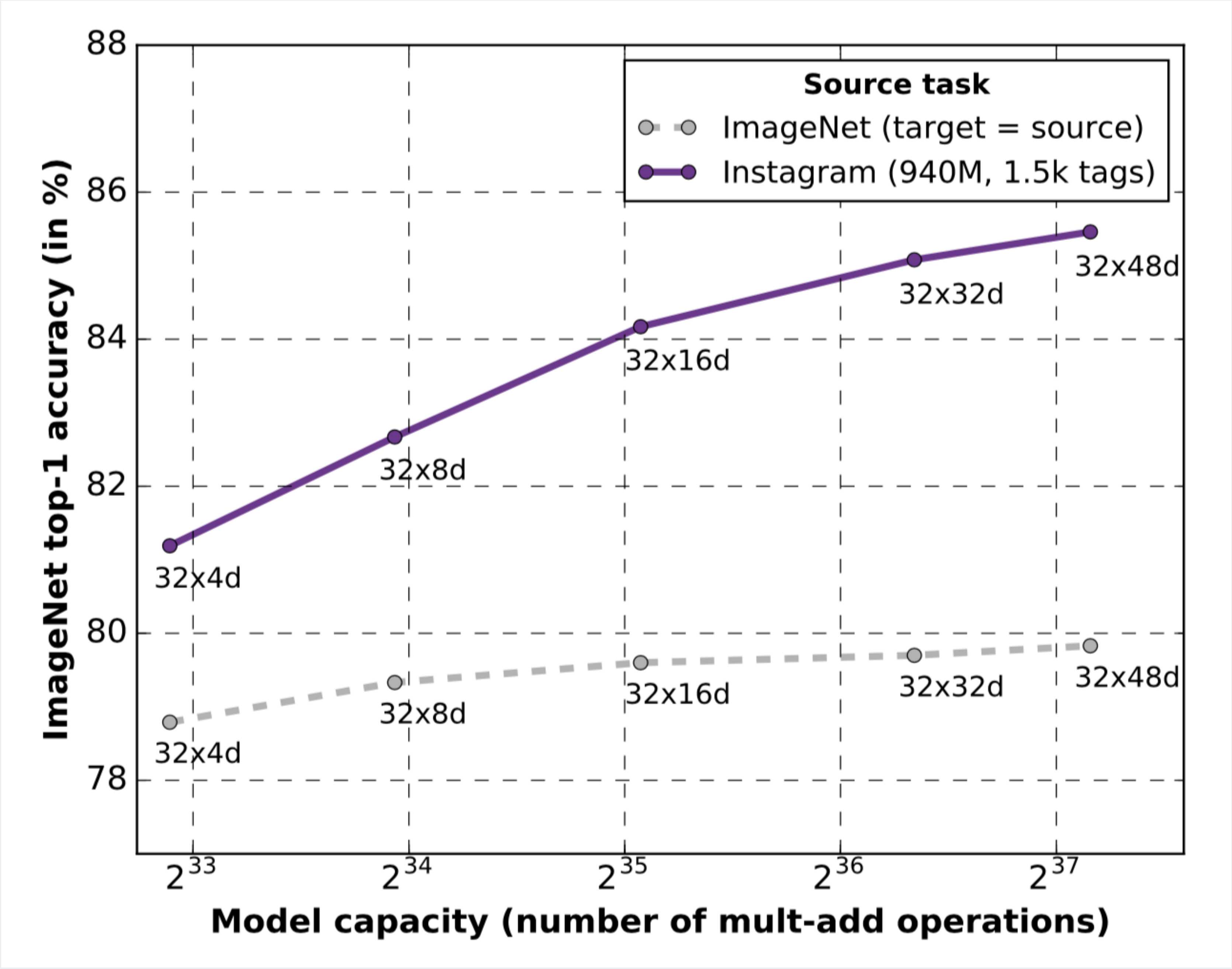
Efficient AI inference challenges

- Capacity crunch
- Realtime model serving efficiency
- Scale to billions of users

Increase of server capacity by Xiaodong Wang



Accuracy vs Capacity



Agenda

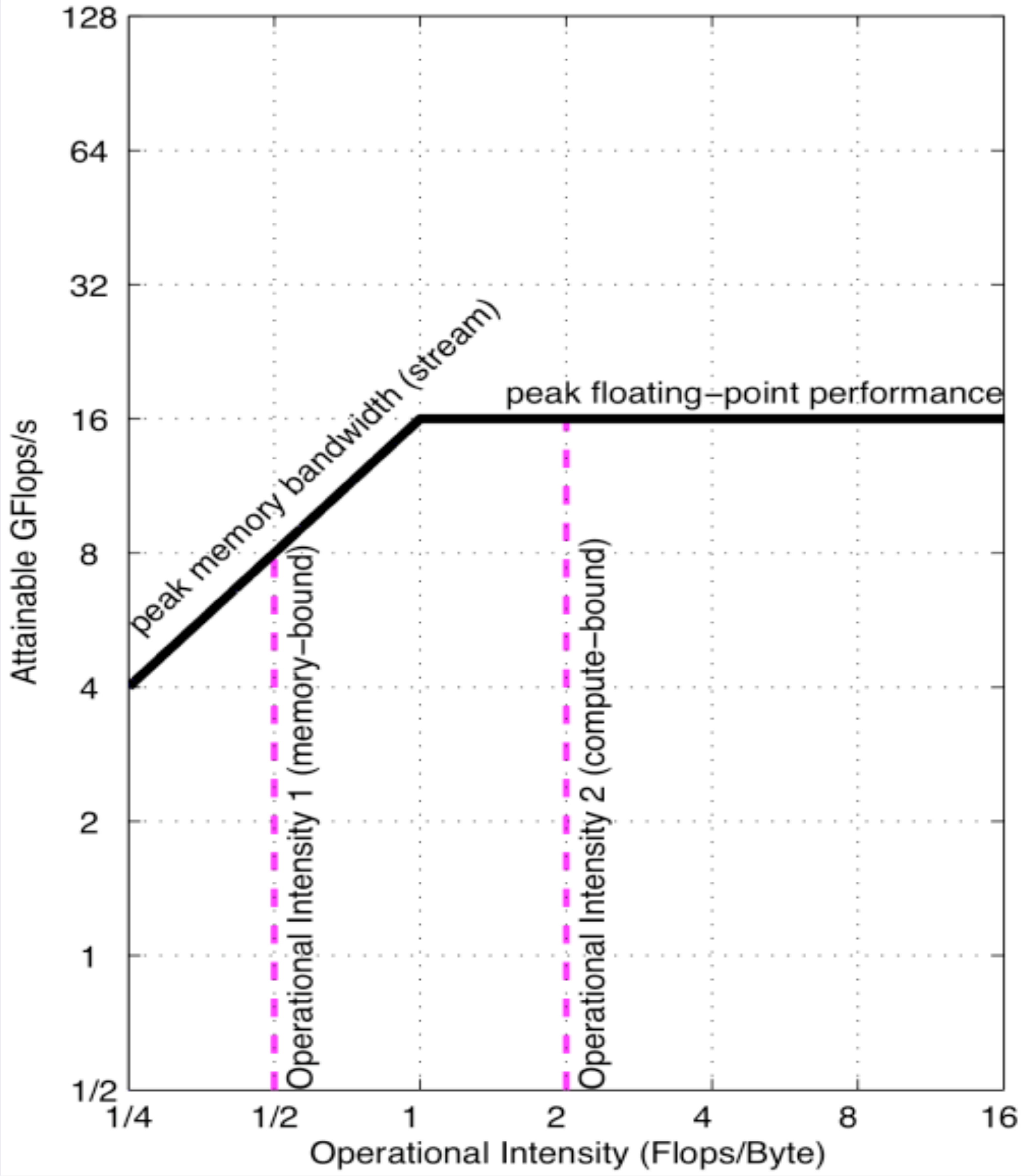
- Facebook AI workload characteristics
- **Low precision computing**
  - Reduced precision floating point optimization
  - Fixed point quantization
- AI system co-design for low precision computing
  - Model co-design
  - Hardware co-design

Low-precision computing

- Default precision: fp32
- Reduced-precision floating point
  - Fp16, bf16, fp8, etc.
- Fixed point quantization
  - Int8, int4, etc.
- Others
  - Posits (Gustafson 2016)
  - Logarithmic, k-means, etc.

Example of reduced precision representations

Performance modeling



Roofline: An Insightful Visual Performance Model for Floating-point Programs and Multicore Architecture. Williams et al.

Given FC (m, n, k), assume T = max(cpu_t, mem_t)

- cpu_t = 2 * m * n * k / C
- mem_t = S * (m * n + m * k + n * k) / B

System performance is:

- memory bandwidth bound when cpu_t <= mem_t;
- Otherwise, compute bound.

Compute bound scenarios:

- CV

Memory bound scenarios:

- Language translation, recommendation

Reduced precision optimizations

- Fp16:
  - Good programmability and negligible accuracy loss
- Use cases:
  - Prepack the weights in NNs into fp16
  - Convert dense and sparse features to fp16 for end-to-end performance optimizations

Recommendation systems

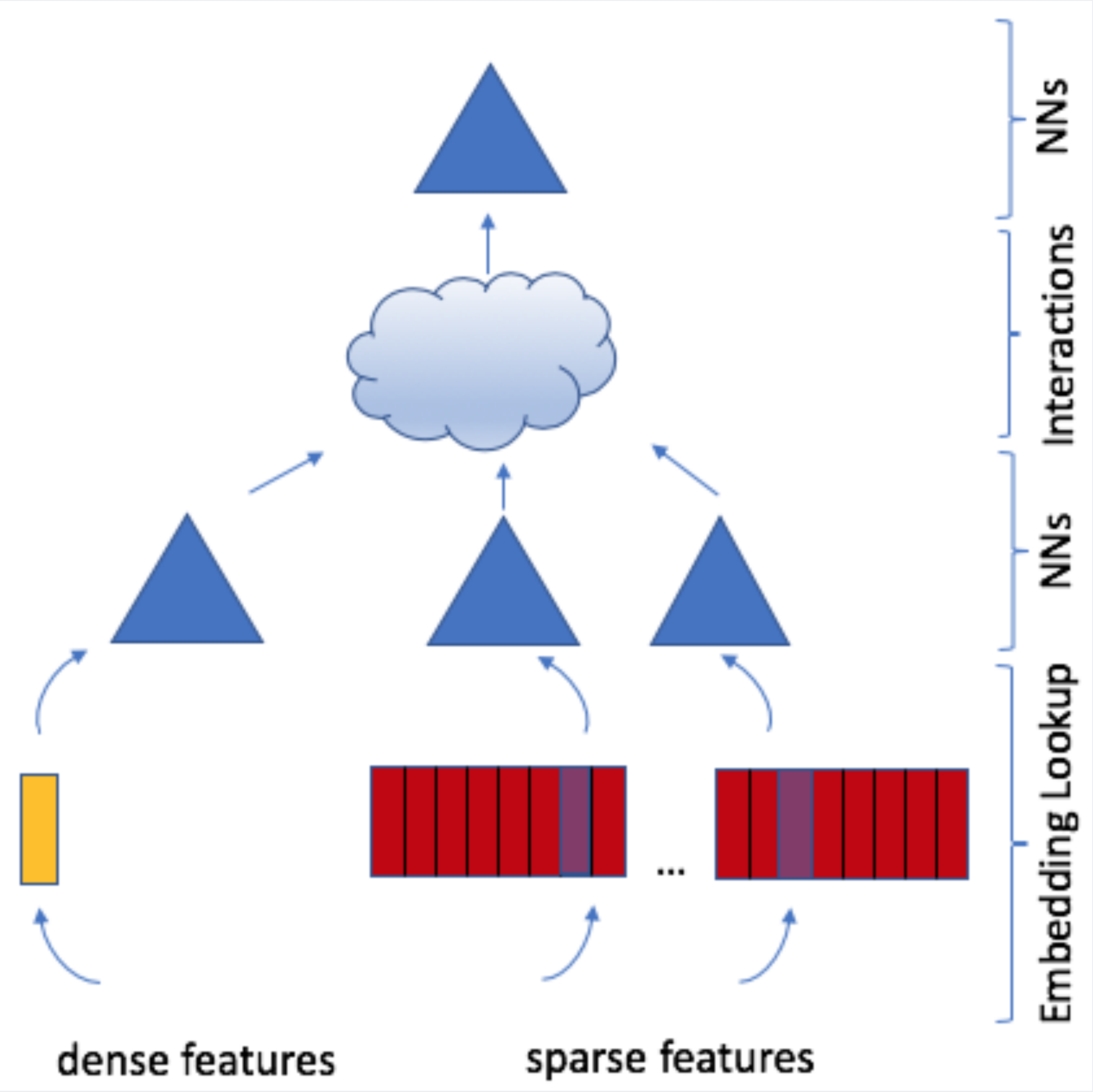
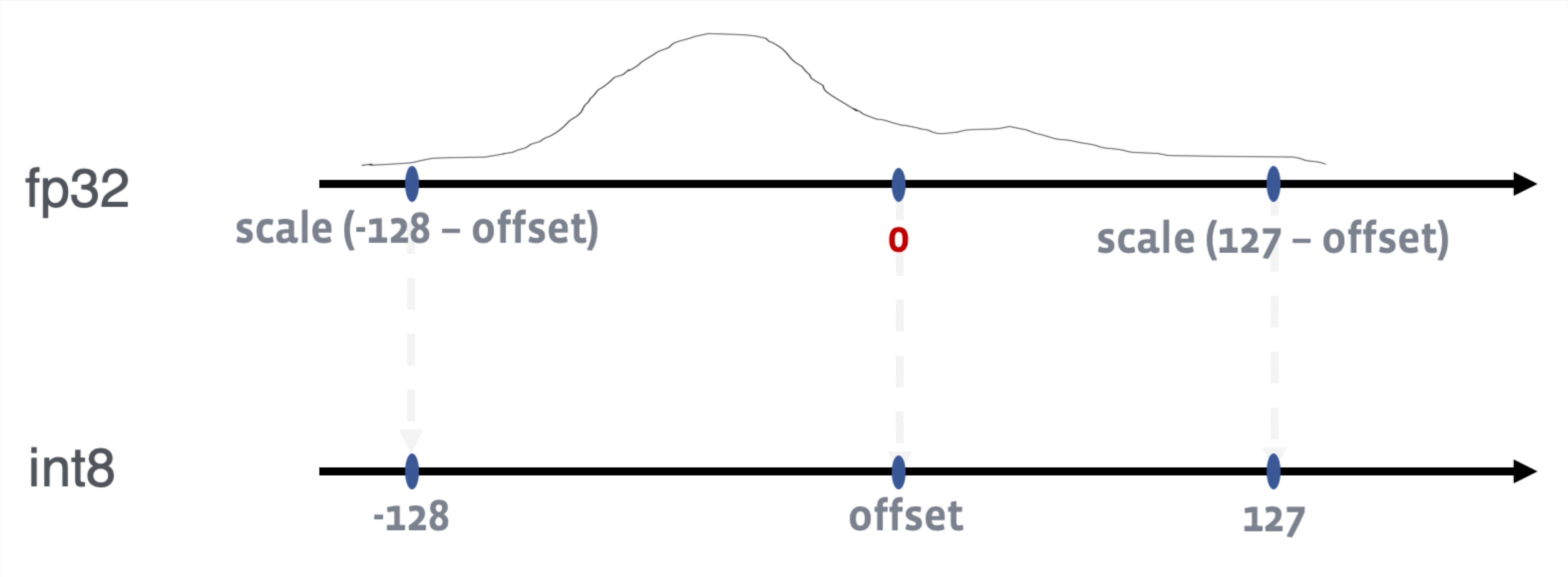


Figure credit: Maxim Naumov

Int8 quantization

- Dequantization: $x = scale \cdot (x_q - offset)$
- Quantization: $x_q = clip(round(x/scale) + offset, -128, 127)$



Challenges

- Accuracy requirements
  - 0.02% for recommendation systems
  - 0.5% for computer vision models
- Performance optimizations

Accuracy improving techniques (1)

- Symmetric vs. Asymmetric
  - preserve sparsity, no nasty handling of offsets during matmul
  - slight loss of accuracy if using int8 for both weights and activations
- Unsigned vs. Signed
- Including 0 or not
- Channel-wise quantization
  - Assign scale and offset for each channel

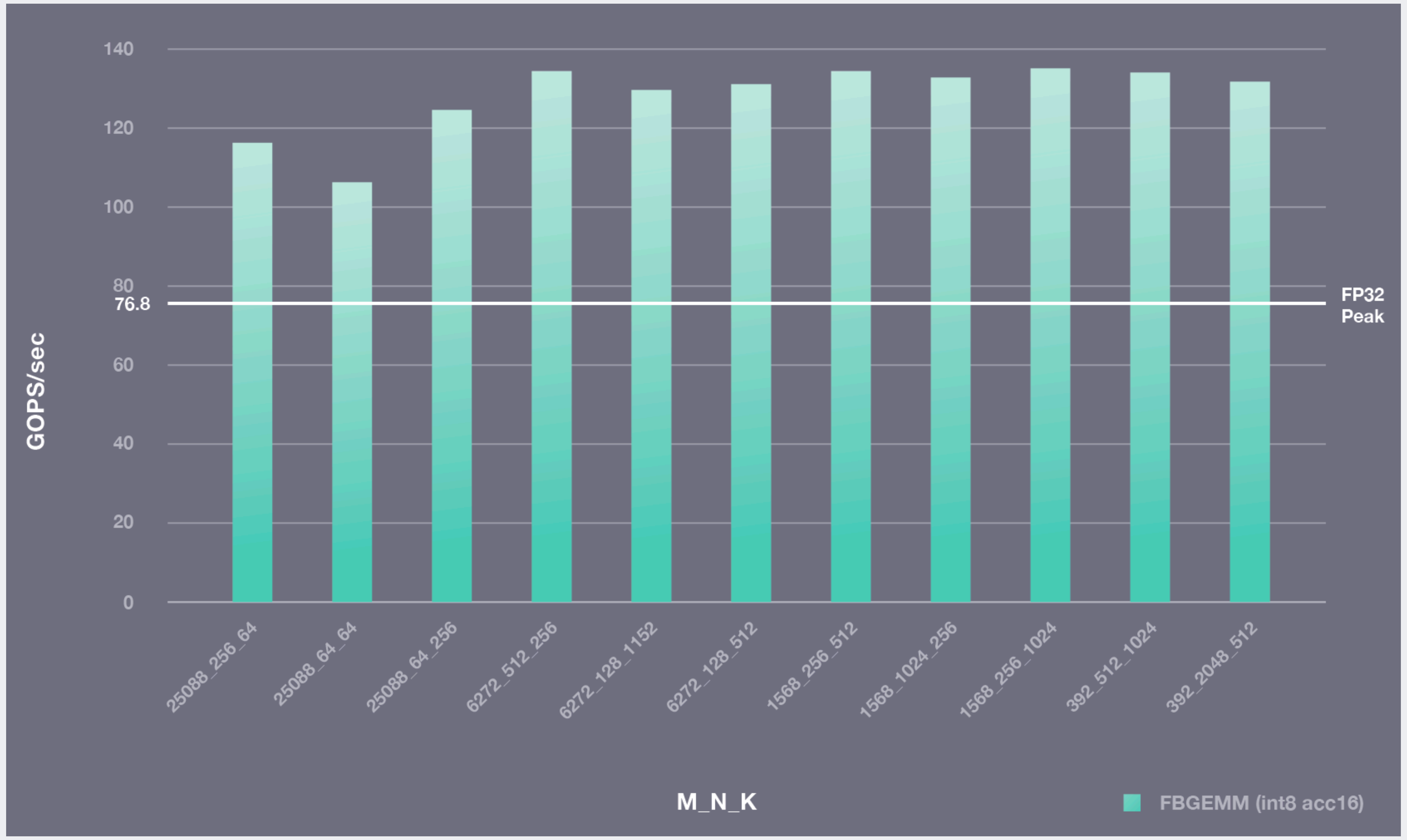
Accuracy improving techniques (2)

- L2 error minimization vs. Min-max
  - Minimize the quantization errors for the more common values while allowing relatively large errors for outliers.
  - Requires the activation histogram profiling offline.
- Outlier-aware quantization

FBGEMM

- Facebook high performance linear algebra library
  - Optimized on-CPU performance for low precision calculations
    - Supports accuracy-loss-minimizing techniques
    - Dynamically generates matrix-shape specific vectorized code

FBGEMM performance for compute bound scenarios



https://code.fb.com/ml-applications/fbgemm/

Int8 quantization for CV models

- OCR text recognition using Rosetta
  - 2x speedups using int8 and int32 acc.
  - 2x speedups using int8 and int16 acc.
    - Outlier-aware quantization
    - Model adjustments
  - Int8 quantization workflow
    - Activation histogram profiling, graph transformation, kernel optimization, quantization space exploration

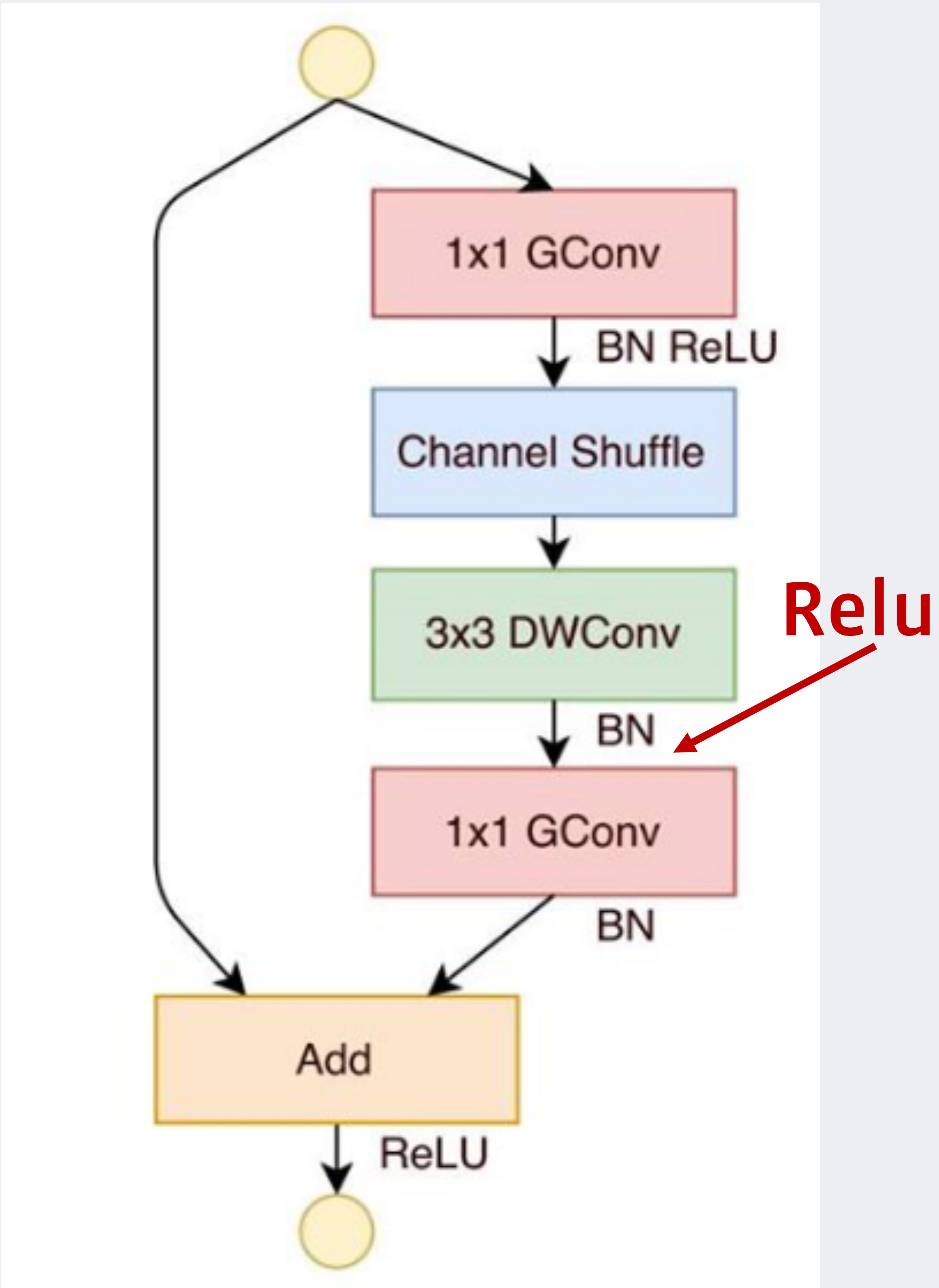
Rosetta: Large scale system for text detection and recognition in images
Fedor Borisyuk et al.

Agenda

- Facebook AI workload characteristics
- Low precision computing
  - Reduced precision floating point optimization
  - Fixed point quantization
- **AI system co-design for low precision computing**
  - Model co-design
  - Hardware co-design

Model co-design

- Int8 quantization on Rosetta
  - + 0.5% accuracy in both fp32 and int8 models
- Int8 quantization on recommendation systems
  - Wider FC layers to compensate for accuracy loss

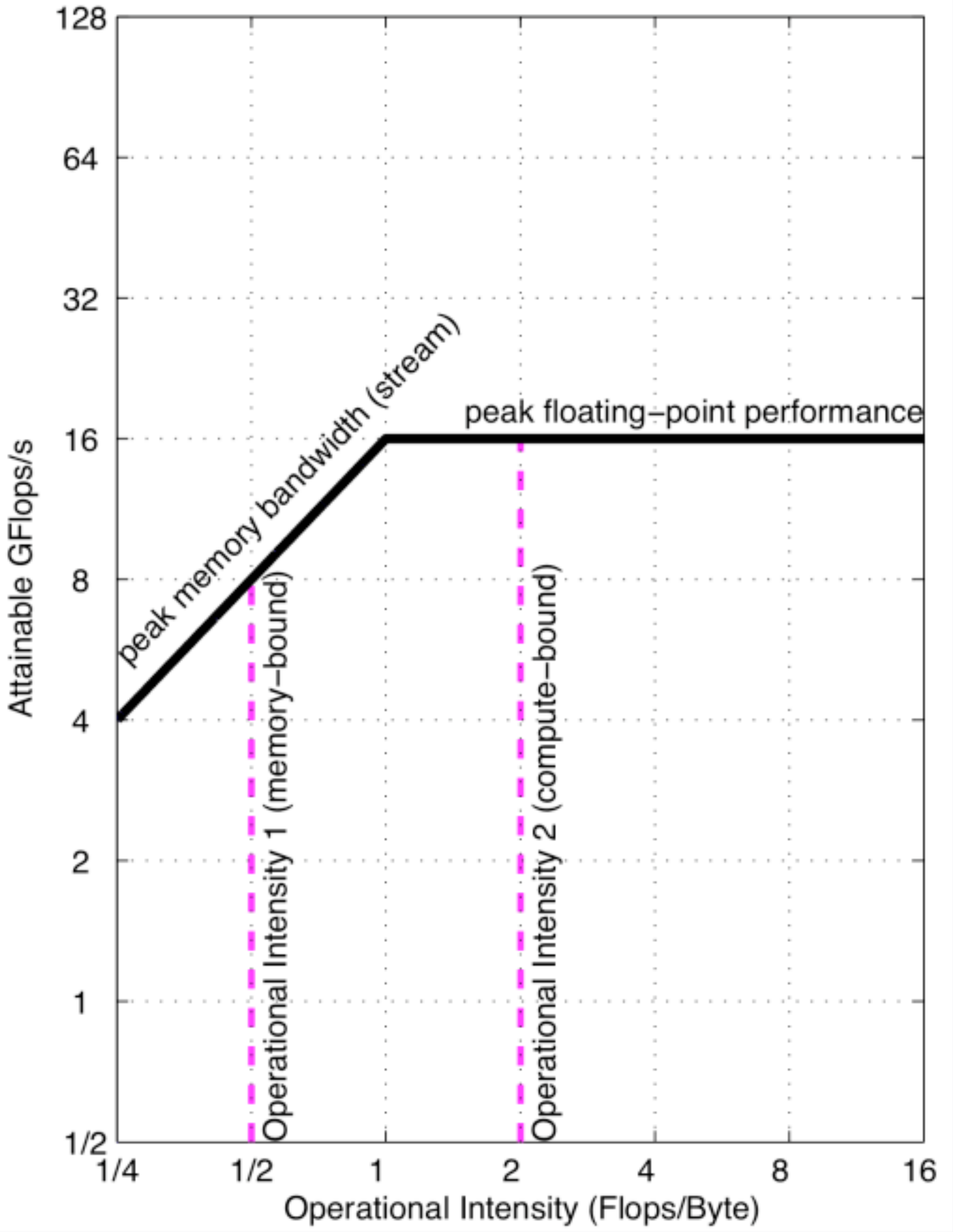


ShuffleNet
https://arxiv.org/pdf/1707.01083.pdf

Hardware co-design

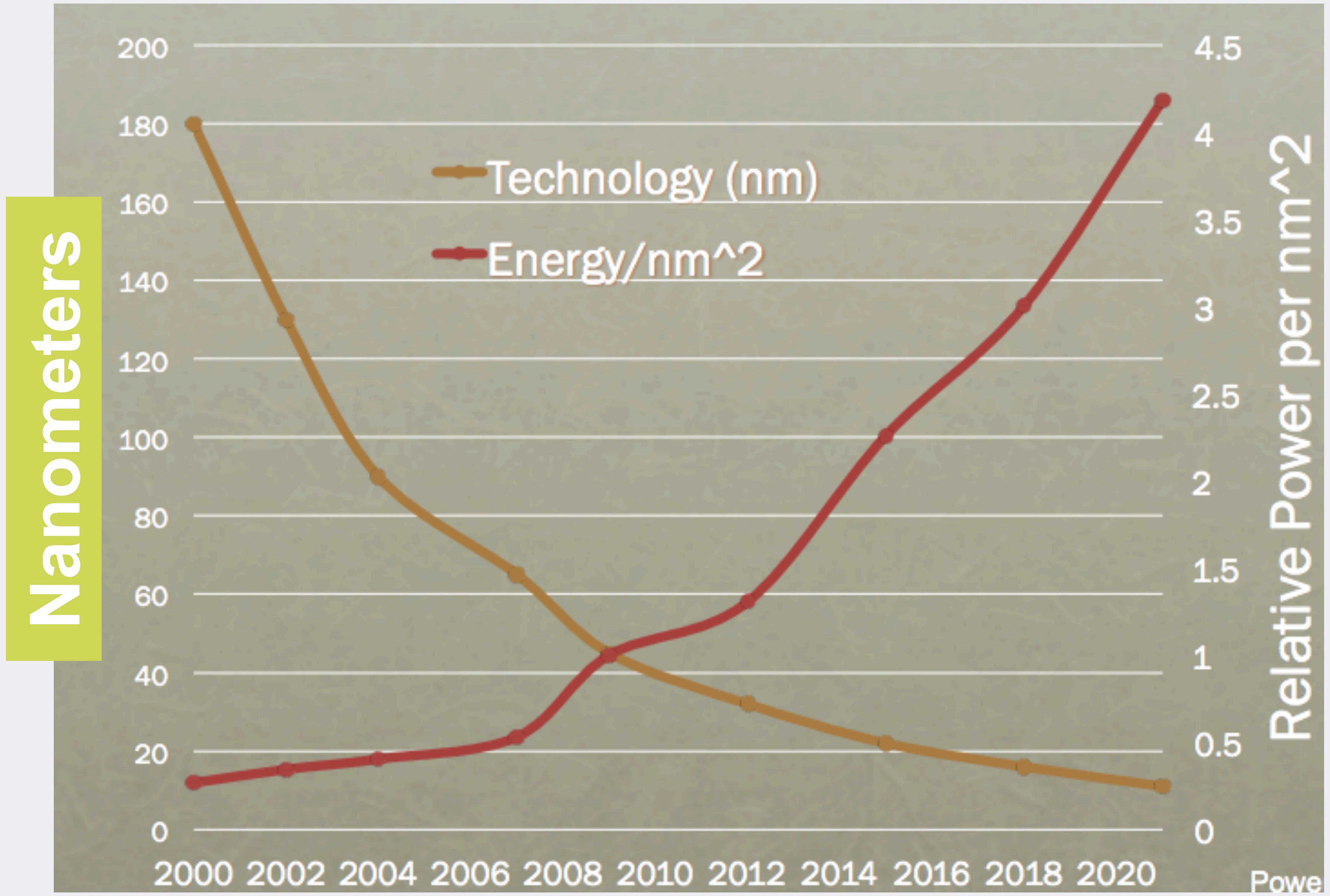
- Low-precision computing can achieve 2x ~ 4x performance improvements on today's hardware

- How to meet the fast growing AI demand for tomorrow?



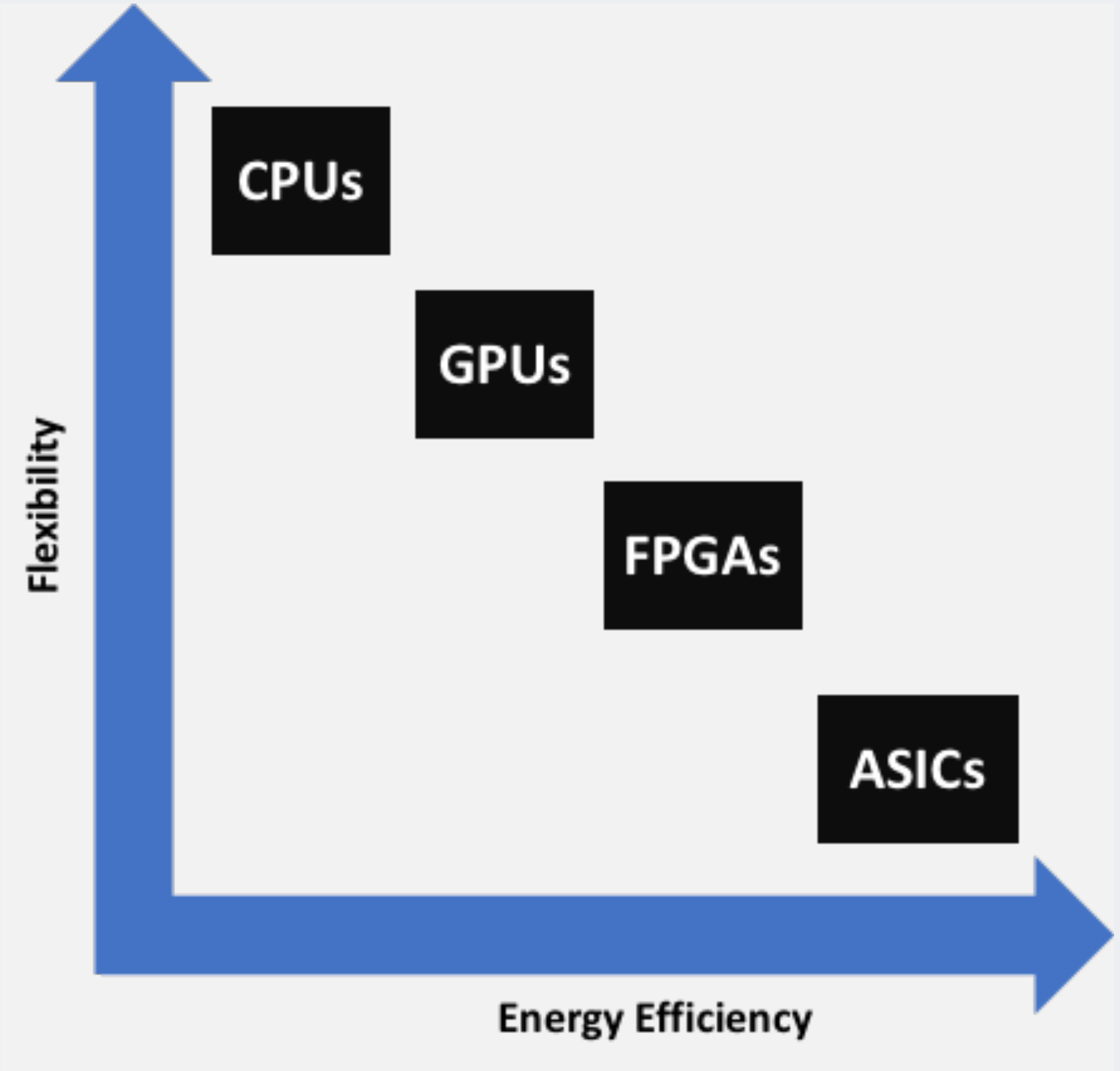
AI Inference Hardware

Technology, energy and Dennard scaling



Future of Computing, John Hennessey

Inference ASIC Hardware



AI Inference Hardware

- Facebook designs its own hardware since 2010
- All designs released through open compute!
- Facebook is partnering with HW vendors to build inference ASIC
- Done via co-design with FB workloads in mind
  - Simulate performance with production models
  - Advise the quantization support from hardware

Thanks!

- Q&A