Multi-Million Core, Multi-Wafer AI Cluster

Cerebras Systems

Sean Lie
Co-founder & Chief Hardware Architect

Cerebras Systems

Building and deploying a new class of computer system

Designed for the purpose of accelerating AI and changing the future of AI work

Founded in 2016

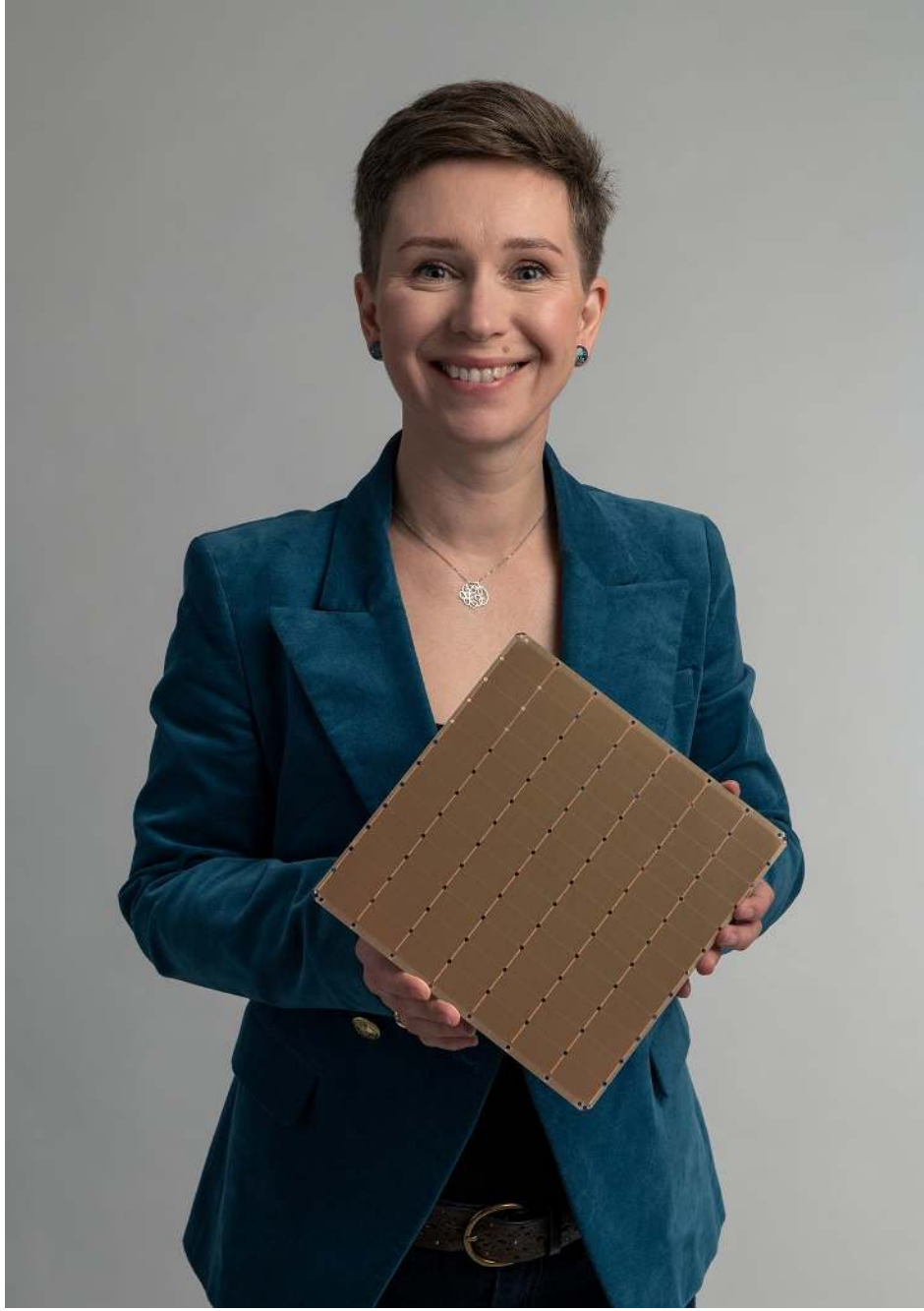
350+ Engineers in 14 Countries

Offices
Silicon Valley | San Diego | Toronto | Tokyo

Customers
North America | Asia | Europe

Cerebras WSE-2
The Largest Chip Ever Built

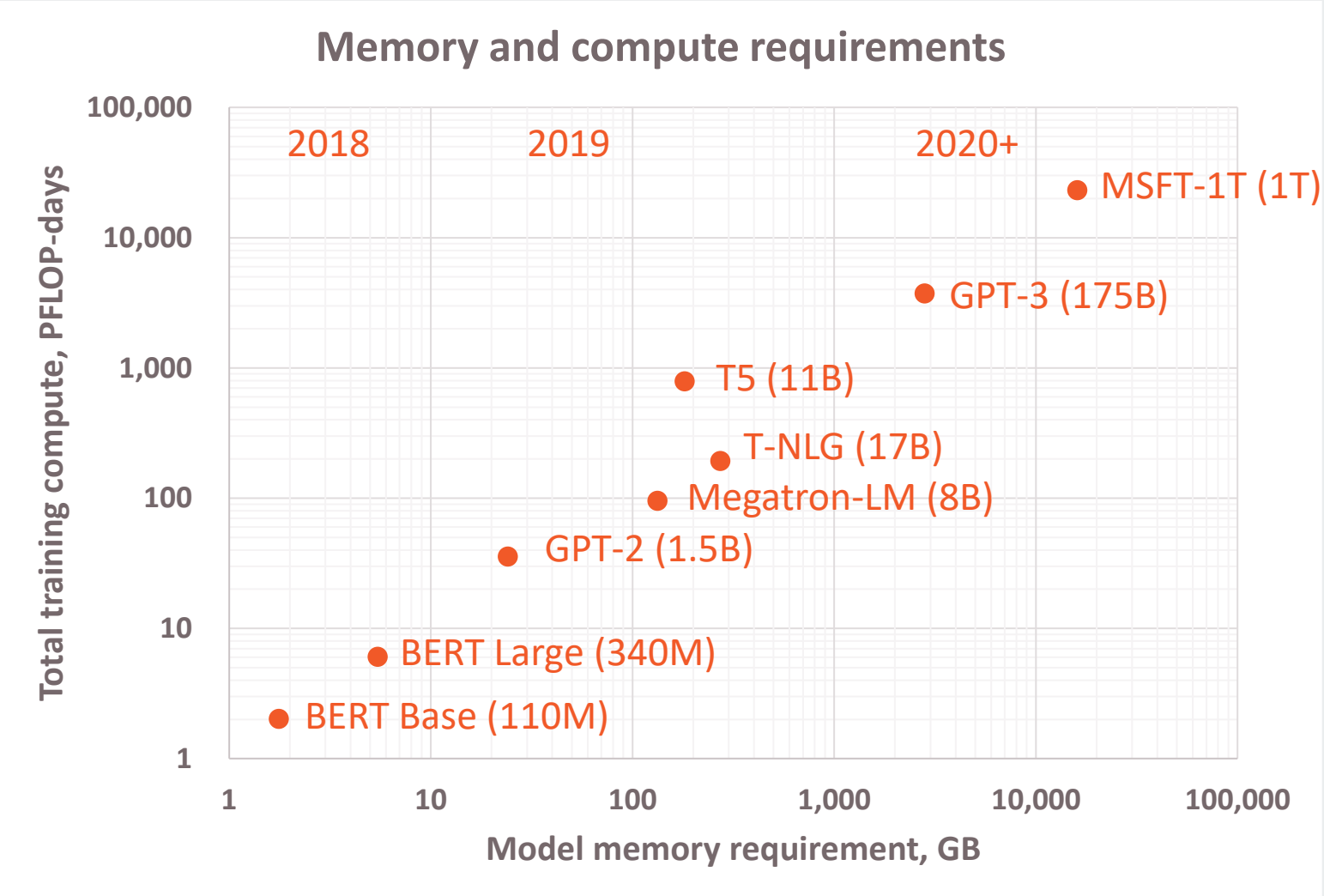
46,225	mm^2 silicon
2.6	Trillion transistors
850,000	AI optimized cores
40	Gigabytes on-chip memory
20	Petabyte/s memory bandwidth
220	Petabit/s fabric bandwidth
7nm	Process technology at TSMC



cerebras

Cerebras CS-2

Exponential Growth of Neural Networks



1000x **larger models**

1000x **more compute**

In just **2 years**

Today, GPT-3 with 175 billion params trained on 1024 GPUs for 4 months.

Tomorrow, **multi-trillion** parameter models and beyond.

Why is this so hard today?

Multi-trillion parameters model need massive **memory**, massive **compute**, and massive **communication**.

On giant clusters of small devices, **all three become intertwined, distributed problems**.

Need to do inefficient, fine-grained partitioning and coordination of memory, compute, and communication across thousands of devices.

Distribution complexity scales dramatically with cluster size

Unlocking Brain-Scale Neural Networks

120-Trillion Parameter Models on a Single CS-2

192 CS-2 Cluster with Near-Linear Performance Scaling

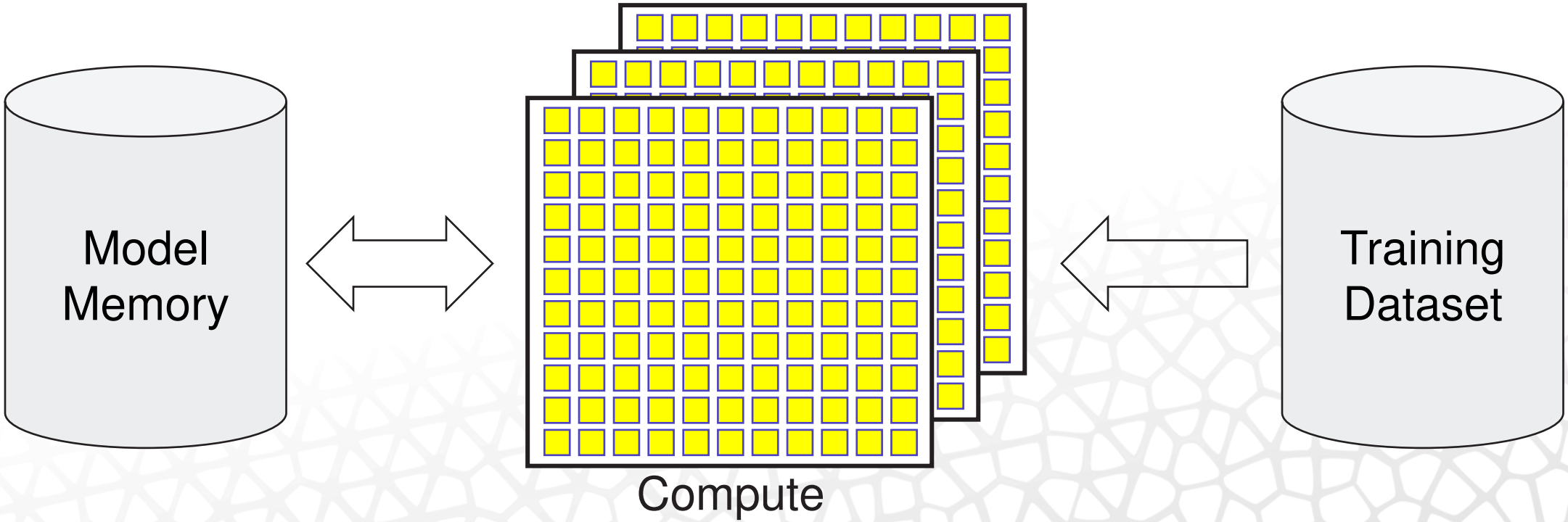
10x Weight Sparsity Speedup

Push-Button Scaling

cerebras

Weight Streaming Unlocks Extreme-Scale Models

A complete disaggregation of memory and compute

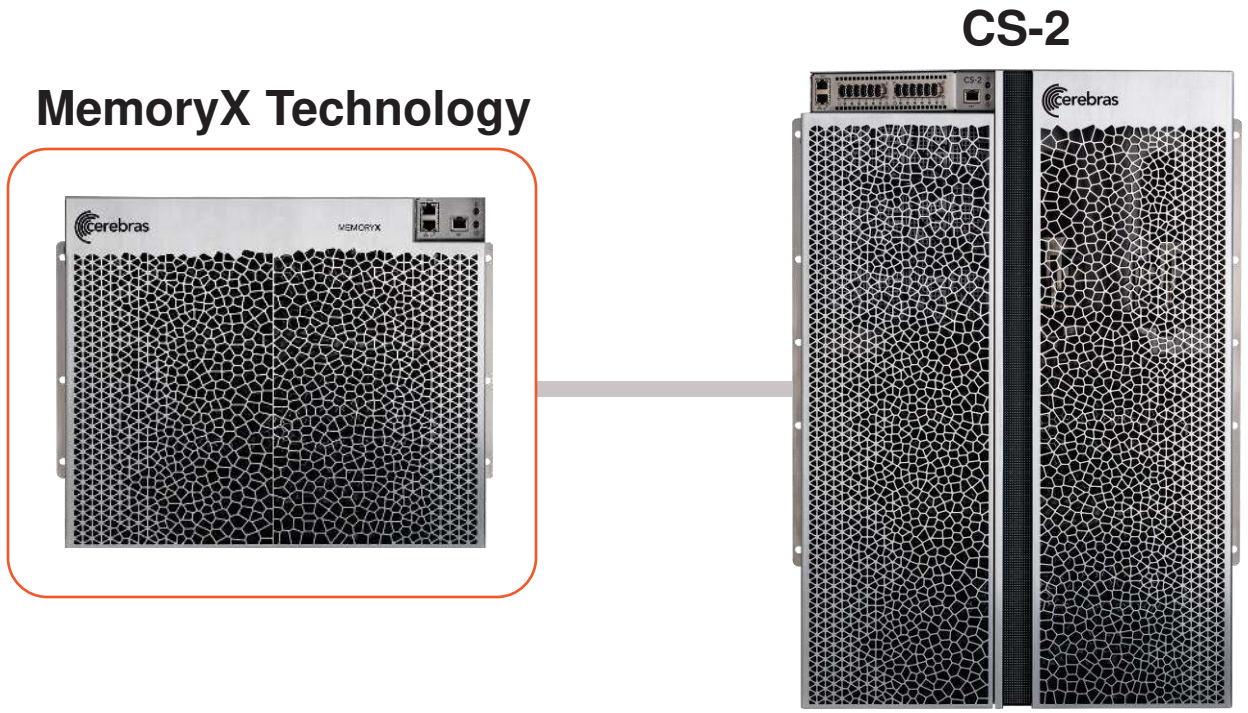


Flexible scaling of model size and training speed

CS-2

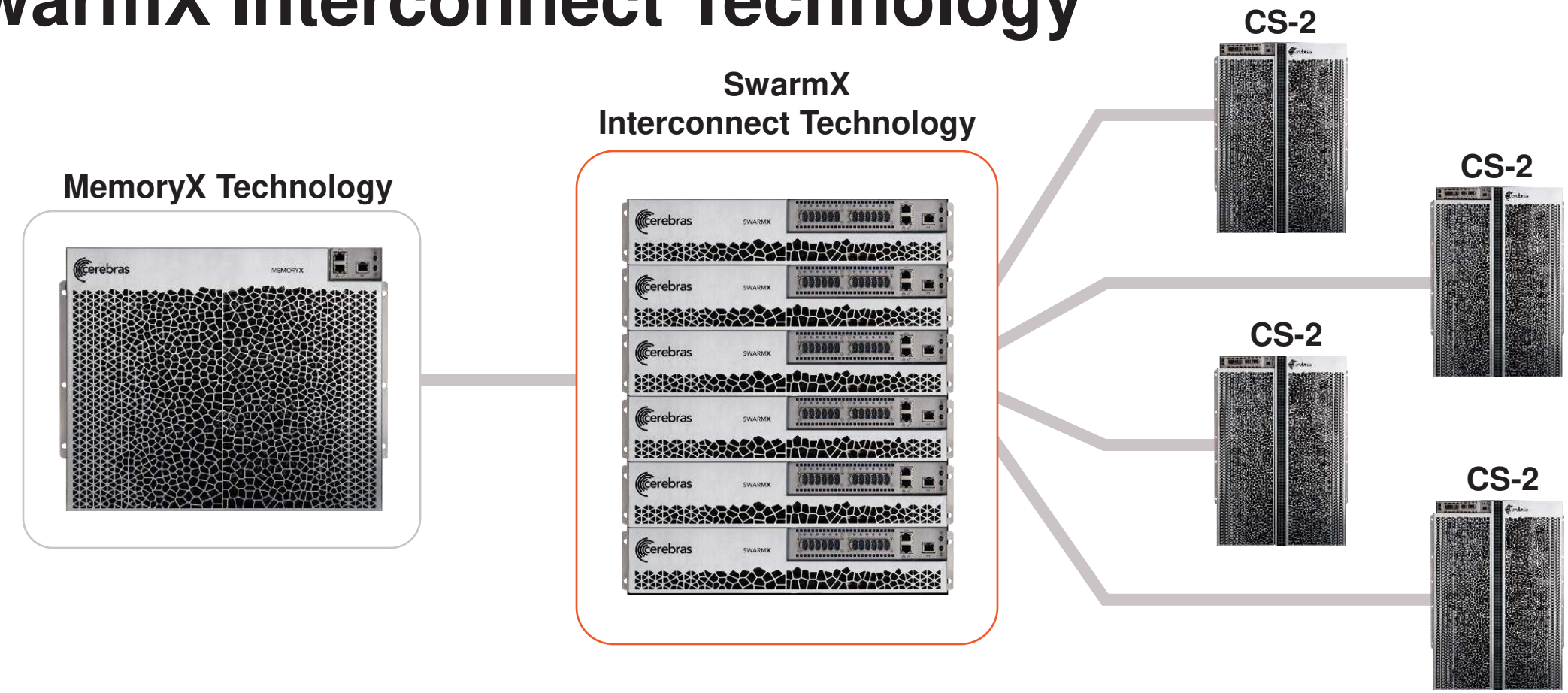
850,000 compute cores in single chip

MemoryX Technology



Up to 120 trillion parameters on a single CS-2

SwarmX Interconnect Technology



Near-linear performance scaling up to 192 CS-2s

Push-button Software Scaling Ease

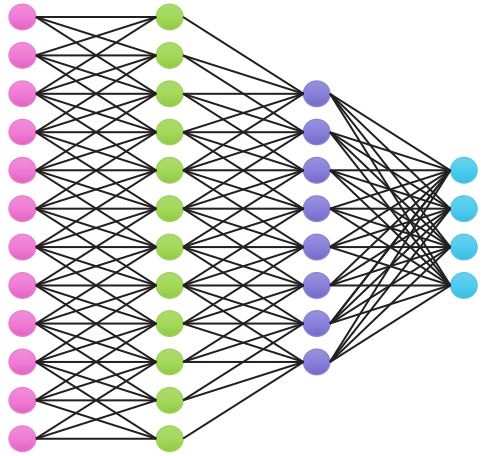
163 million cores, programming ease of a single system

Rethinking the Execution Model

Weight Streaming Execution Model

Built for extreme-scale neural networks:

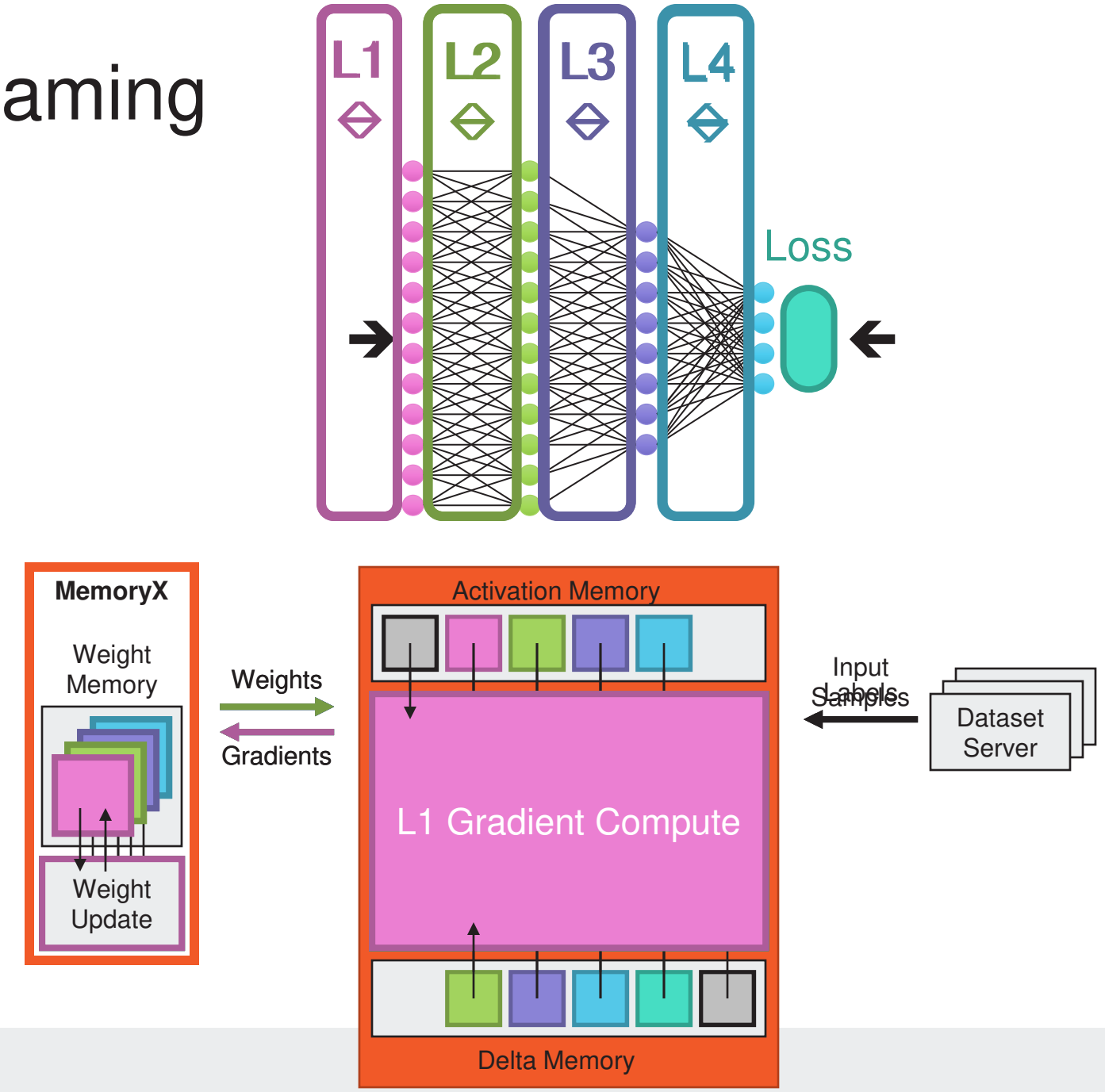
- Weights stored externally off-wafer
- Weights streamed onto wafer to compute layer
- Activations only are resident on wafer
- Execute one layer at a time



Decoupling weight optimizer compute

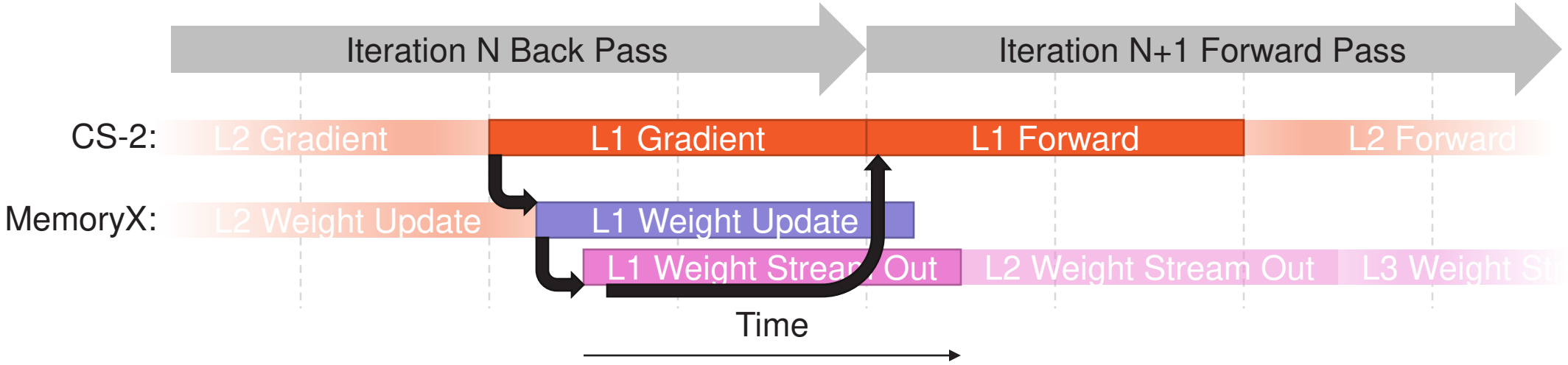
- Gradients streamed out of wafer
- Weight update occurs in MemoryX

Weight Streaming In Action



Solving the Latency Problem

- Efficient performance scaling by removing latency-sensitive communication
- Coarse-grained pipelining
  - Forward/delta/gradient are fully pipelined
  - All streams within an iteration have no inter-layer dependencies
- Fine-grained pipelining
  - Overlapping of weight update and forward pass covers inter-iteration dependency



Extreme Capacity

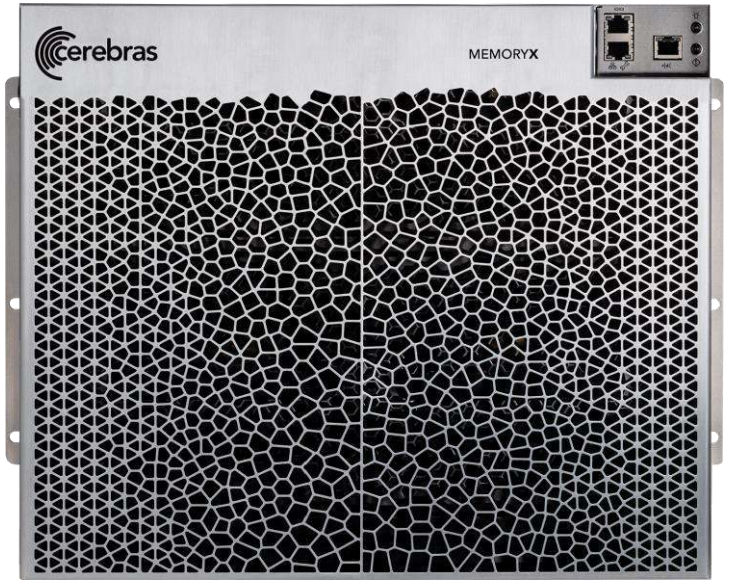
Two capacity problems for extreme-scale models

1. How do you store the giant model?
2. How do you run that giant model on a chip?

MemoryX Technology

Purpose-built to support large neural network execution:

- 4TB – 2.4PB capacity
- 200 billion – 120 trillion weights with optimizer state
- DRAM and flash hybrid storage
- Internal compute for weight update/optimizer
- Handles intelligent pipelining to mask latency



Scalable to extreme model sizes

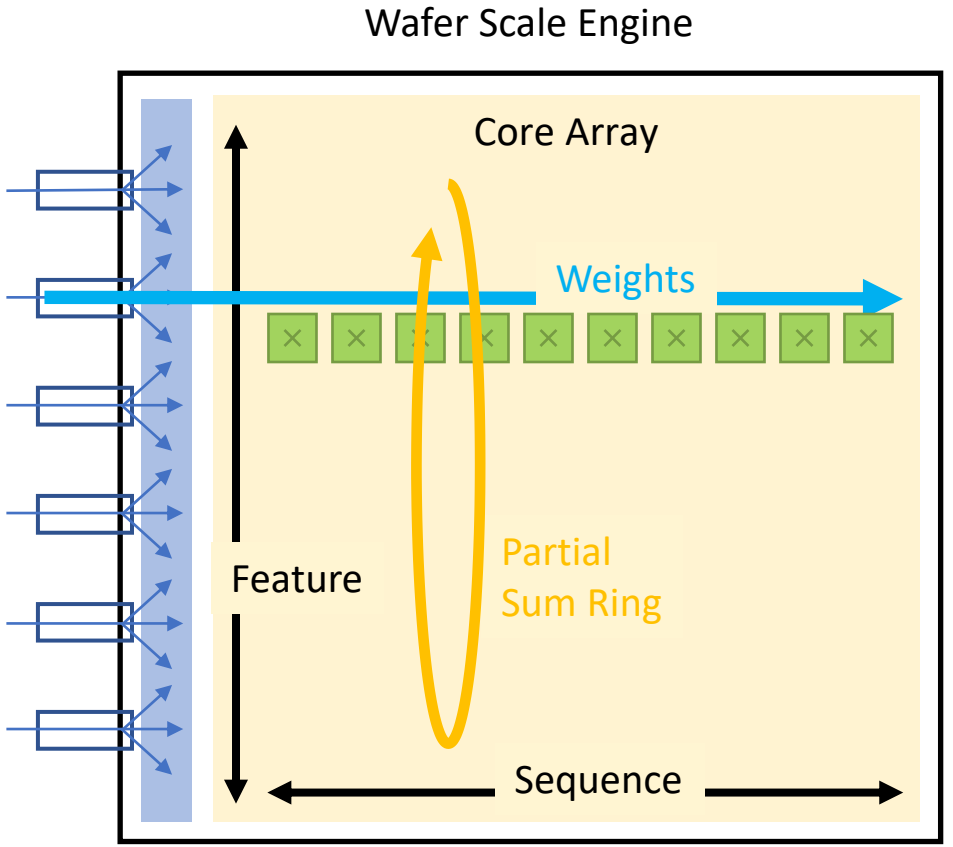
Capacity scaling independent from compute

WSE Enables Extreme-Sized Layers On Chip

A NN layer is almost entirely MatMul and…

The Wafer is the MatMul array

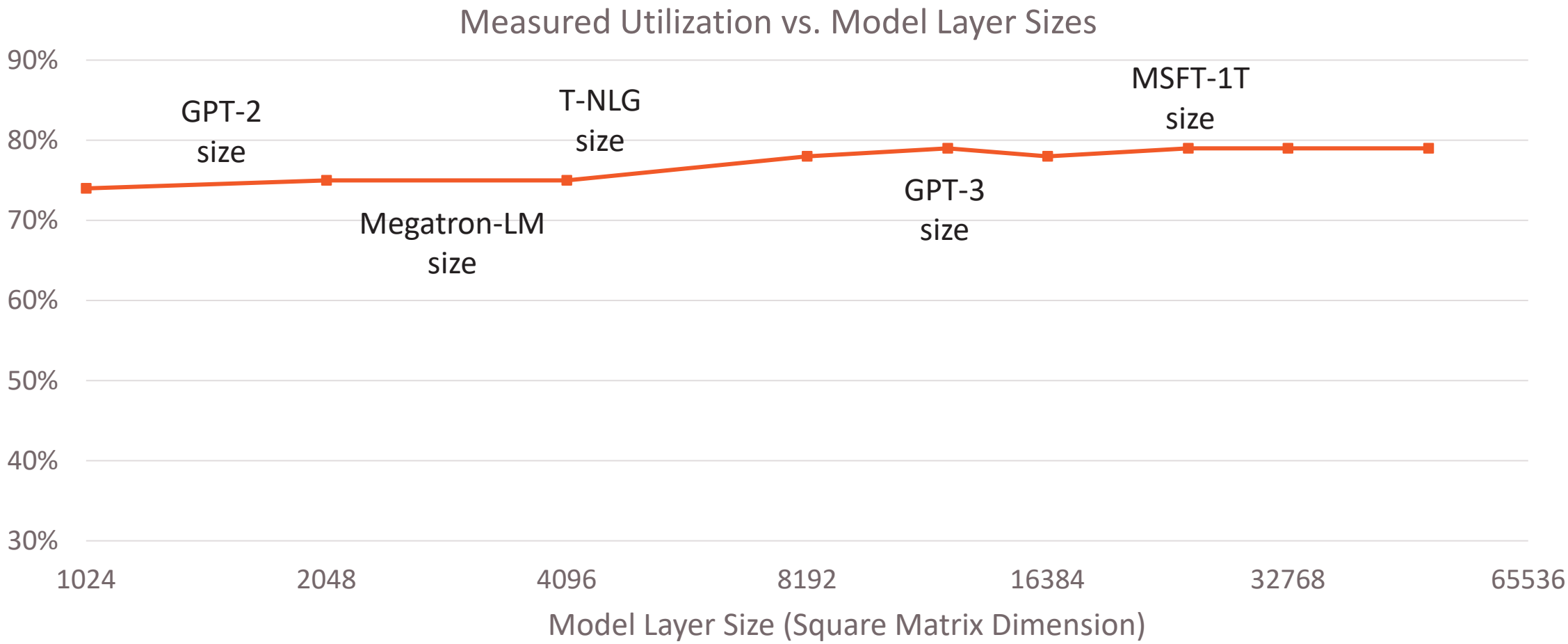
- **High-capacity local memory** stores all activations across compute fabric
- **Large compute core array** receives weight stream and multiplies with activations, weights never stored
- **Massive memory bandwidth** enables full performance of operands to the datapath
- **High BW interconnect** enables partial sum accumulation across wafer at full performance
- **No matrix blocking** or partitioning required



Fits MatMuls up to size 100k*100k

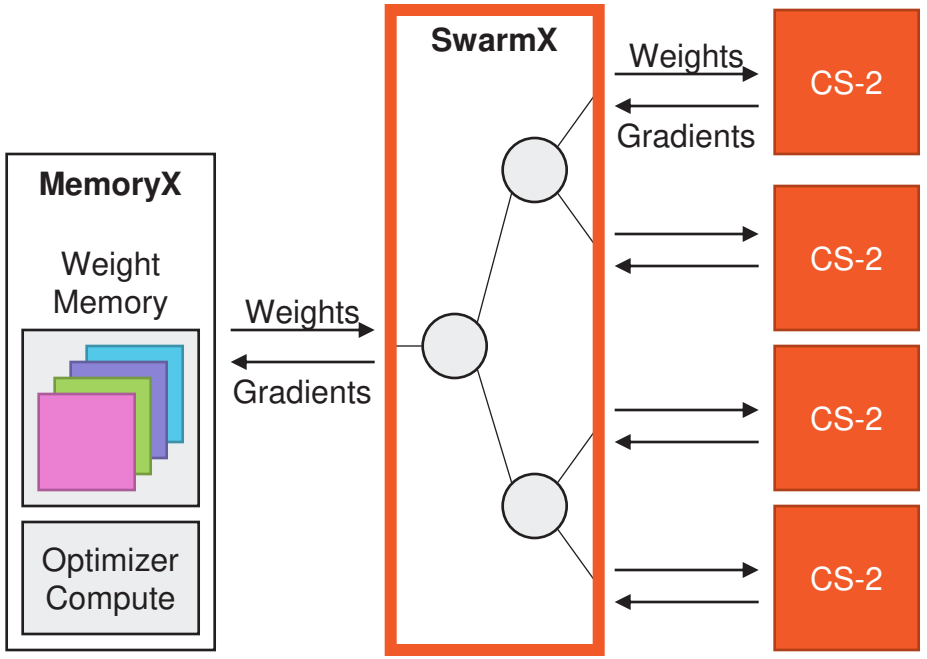
No overhead of splitting MatMul across multiple devices

Demonstrated Scaling to Extreme Layers



Extreme Speed

SwarmX Fabric Connects Multiple CS-2s

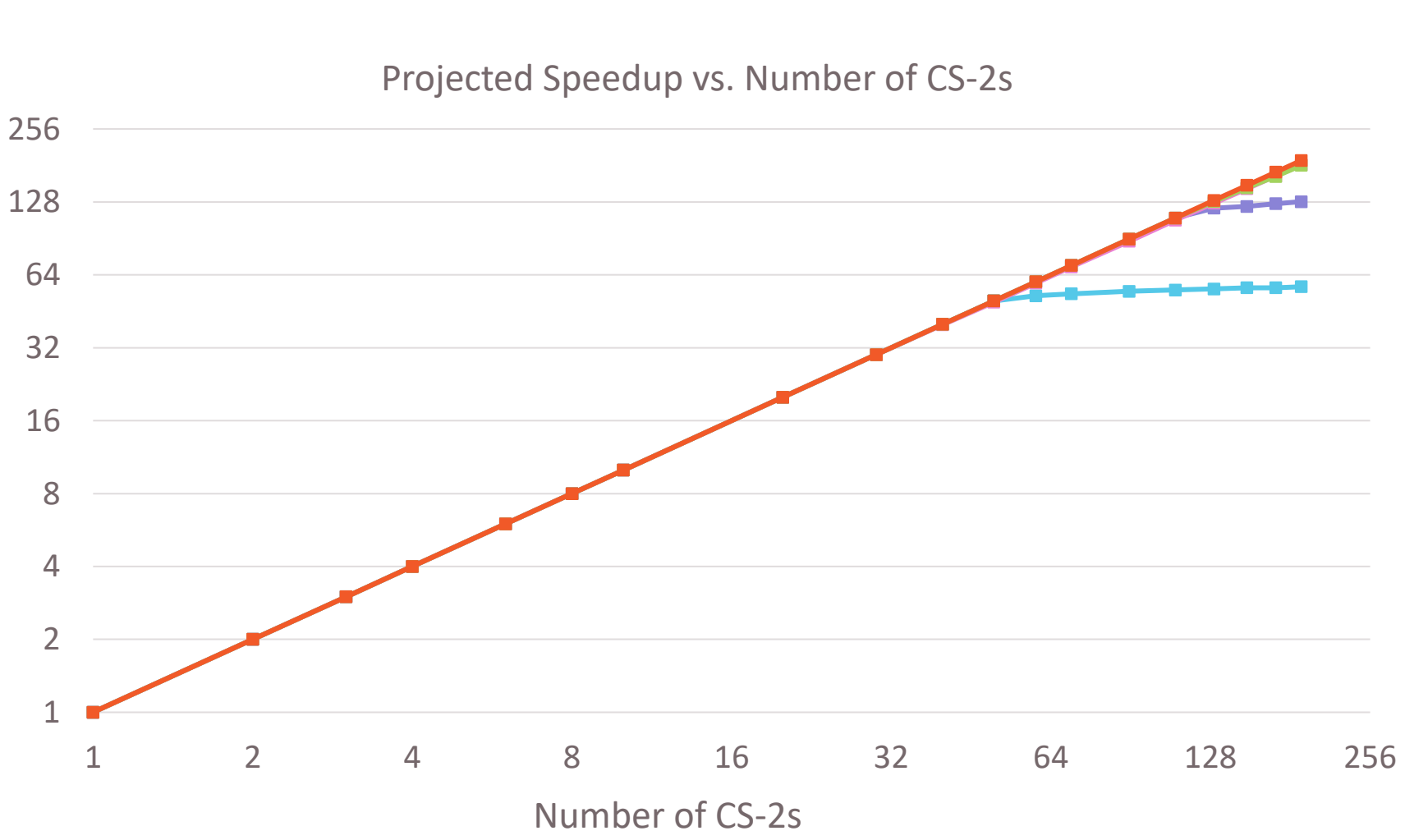


- Data parallel training across CS-2s
- Weights are **broadcast** to all CS-2s
- Gradients are **reduced** on way back
- **Multi-system scaling with the same execution model as single system**
  - Same system architecture
  - Same network execution flow
  - Same software user interface

Scalable to extreme model sizes

Compute scaling independent from capacity

Near-Linear Performance Scaling



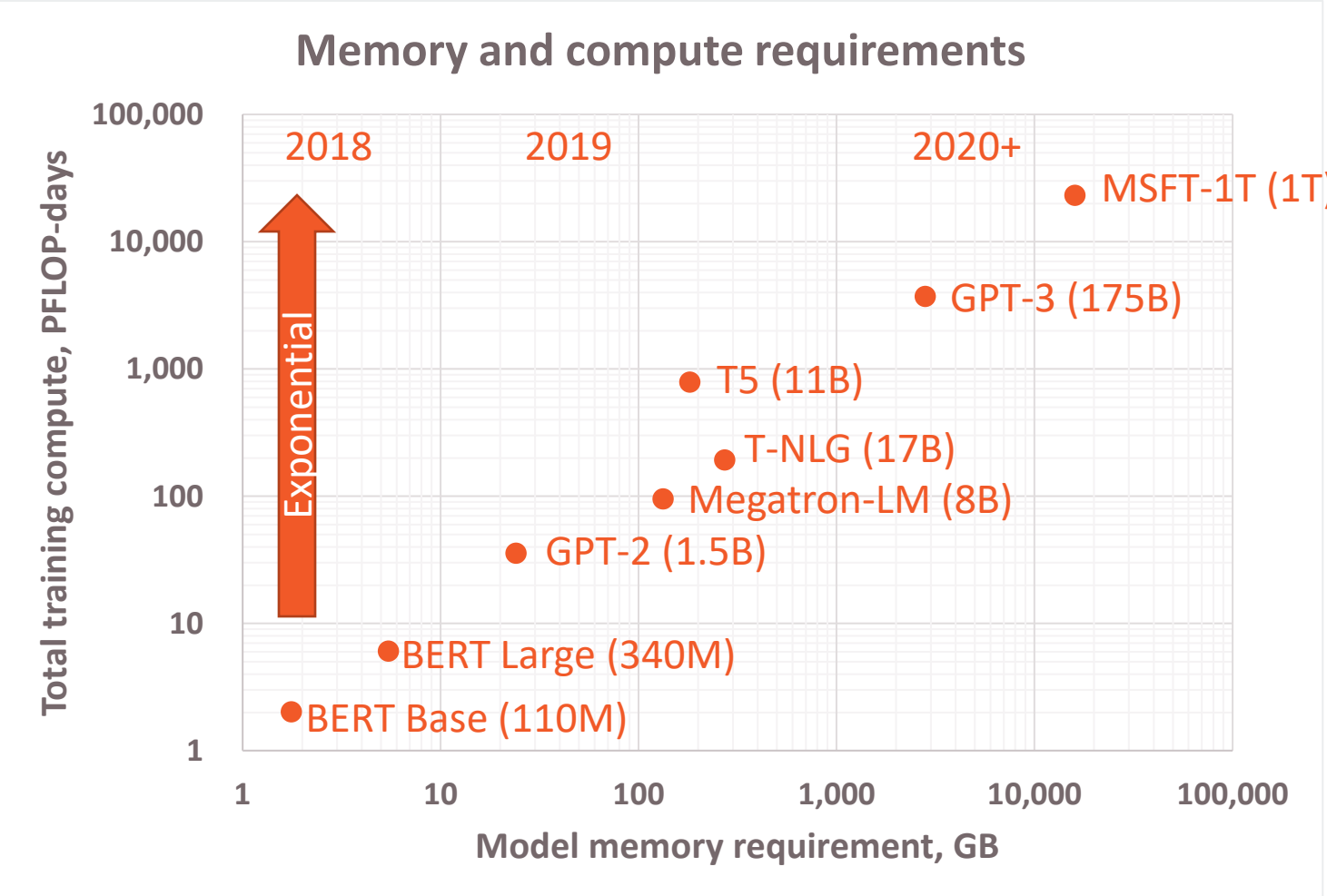
Model	Size
Megatron-LM	8B
T5	11B
T-NLG	17B
GPT-3	175B
MSFT-1T	1T

Projections based on *Scaling Laws for Neural Language Models* [OpenAI]

Extremely Smart

Brute Force Scaling is Not Enough

We Need Faster Training with Less Compute



- Brute-force scaling is the historical path to larger models
- We need it.
- But we also need more.

Algorithmic efficiency lets us use FLOPs more effectively.

Existing Sparsity Research Shows 10x+ Opportunity

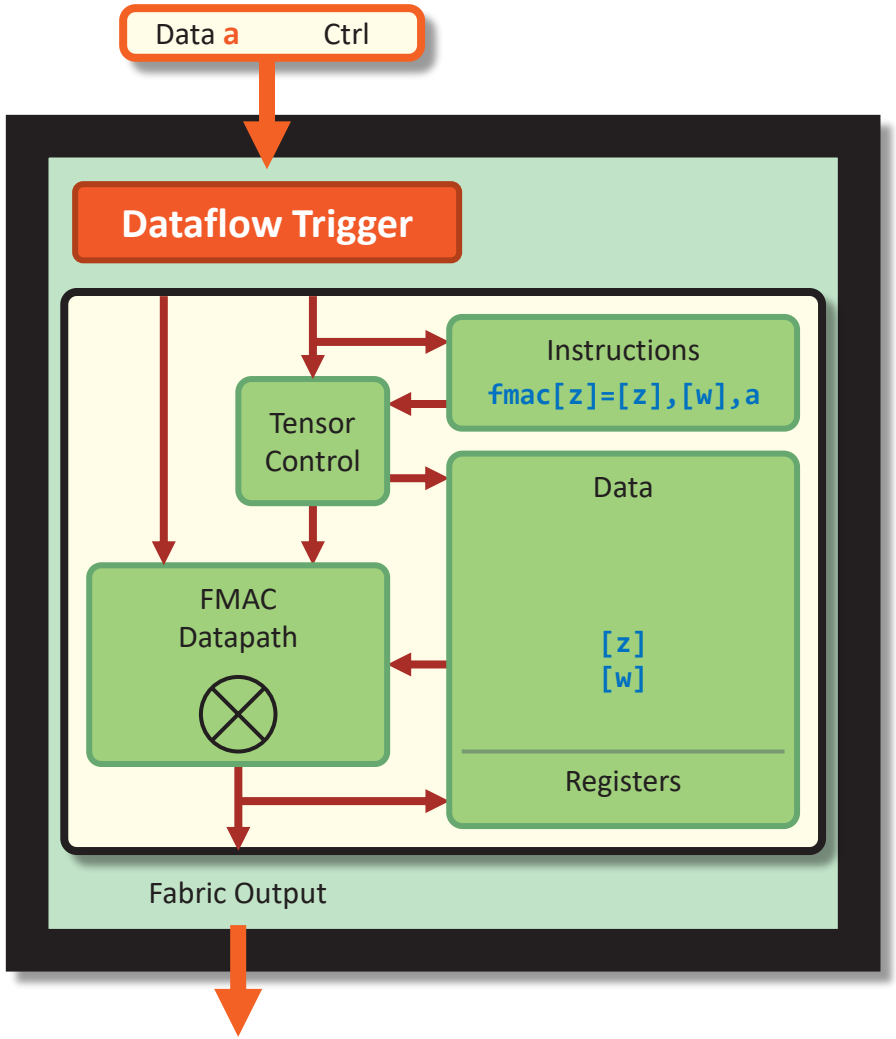
Technique	Sparsity	FLOP ↓	Reference
Fixed Sparse Training	90%	**8x**	Lottery Ticket [MIT CSAIL]
Dynamic Sparse Training	80%	**2x**	Rig the Lottery [Google Brain, DeepMind]
Scaling-up Sparse Training	90%+	**10x+**	Pruning scaling laws [MIT CSAIL]
Monte Carlo DropConnect	50%	**2x**	DropConnect in Bayesian Nets [Nature]

ML Community has invented various sparsity techniques

Cerebras Architecture is Designed for Sparse Compute

- Fine-grained dataflow cores
  - Triggers compute only for non-zero data
- High bandwidth memory
  - Enables full datapath performance
- High bandwidth interconnect
  - Enables low overhead reductions

Only architecture capable of accelerating **all types of sparsity**, including dynamic and unstructured sparsity.



Full Performance on All BLAS Levels

BLAS-3

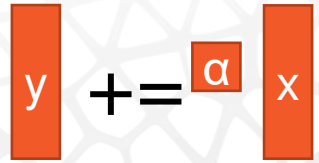
GEMM

BLAS-2

GEMV

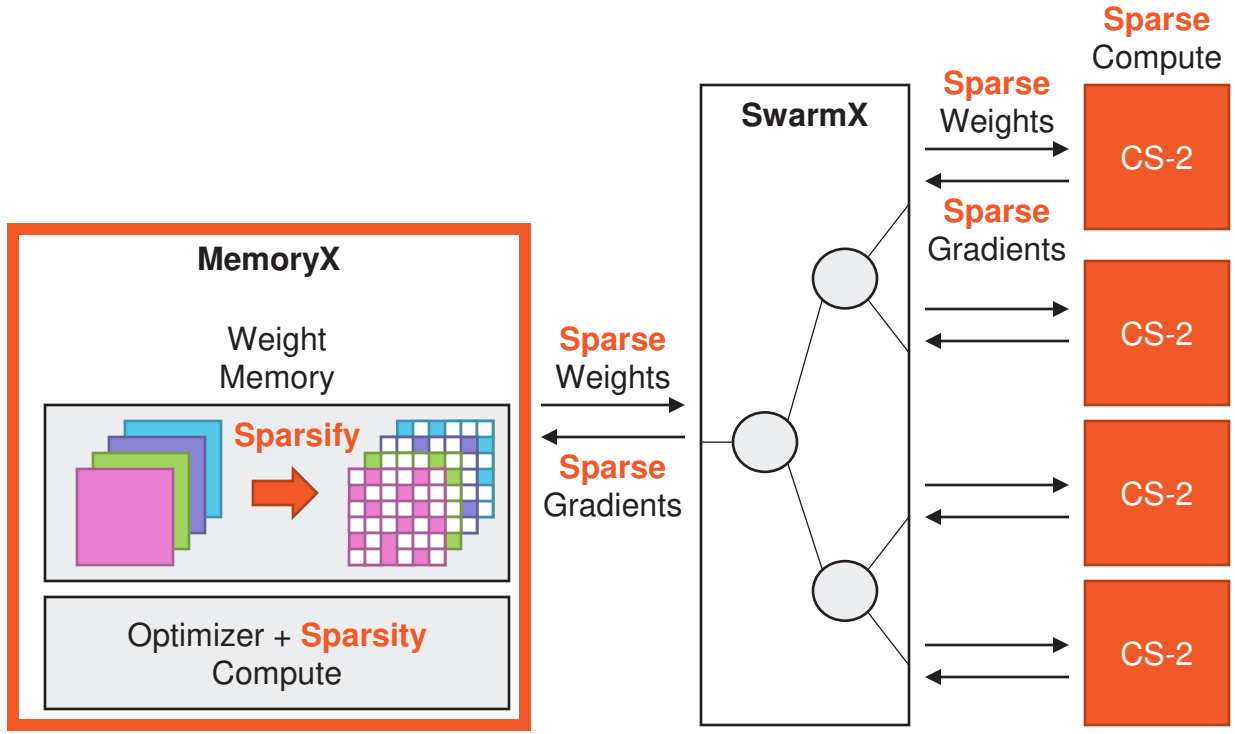
BLAS-1

AXPY



Sparse GEMM is one AXPY per non-zero weight

Streaming Sparse Weights



Weight sparsity induced in MemoryX

- **Sparse weights** streamed to all CS-2s
- **Sparse gradients** reduced on the way back
- **Sparse weight updates** on sparse matrix

No change to the weight streaming model

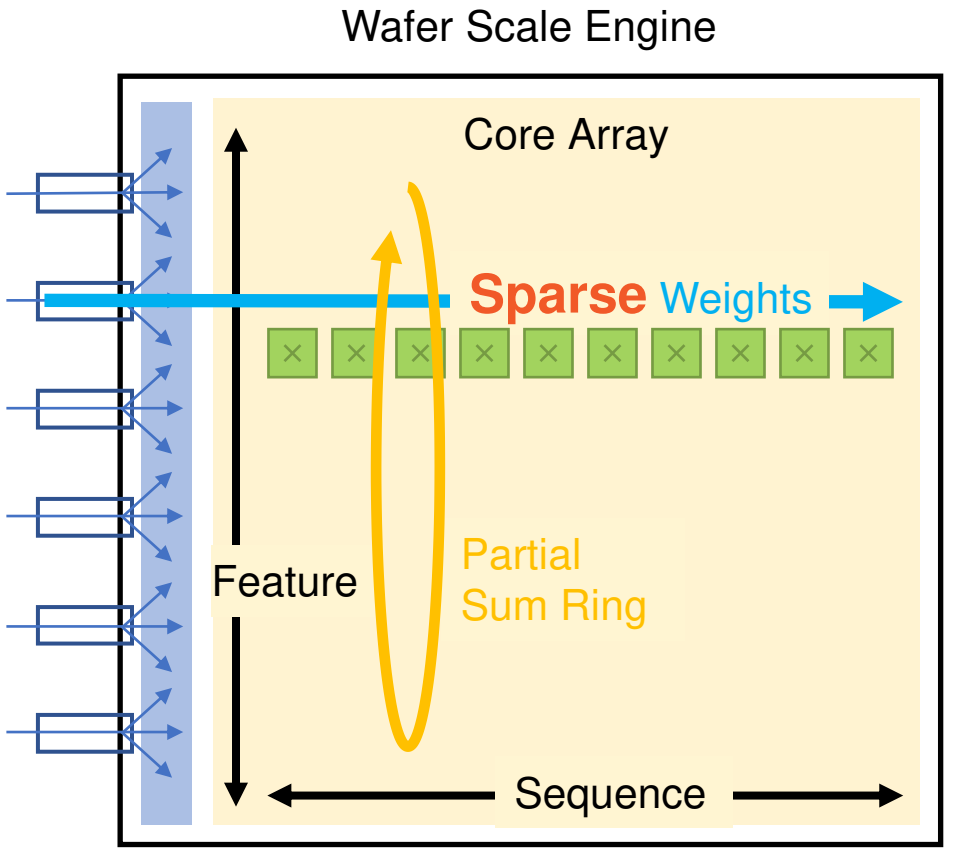
Same flow supports dense and sparse

Native Sparsity Harvesting in Wafer

The Wafer is the Sparse MatMul array

- Local memory stores dense activations across compute fabric
- Core array receives sparse weight stream with only non-zero weight
- Each core performs AXPY with activations, one weight at a time

Same flow supports dense and sparse

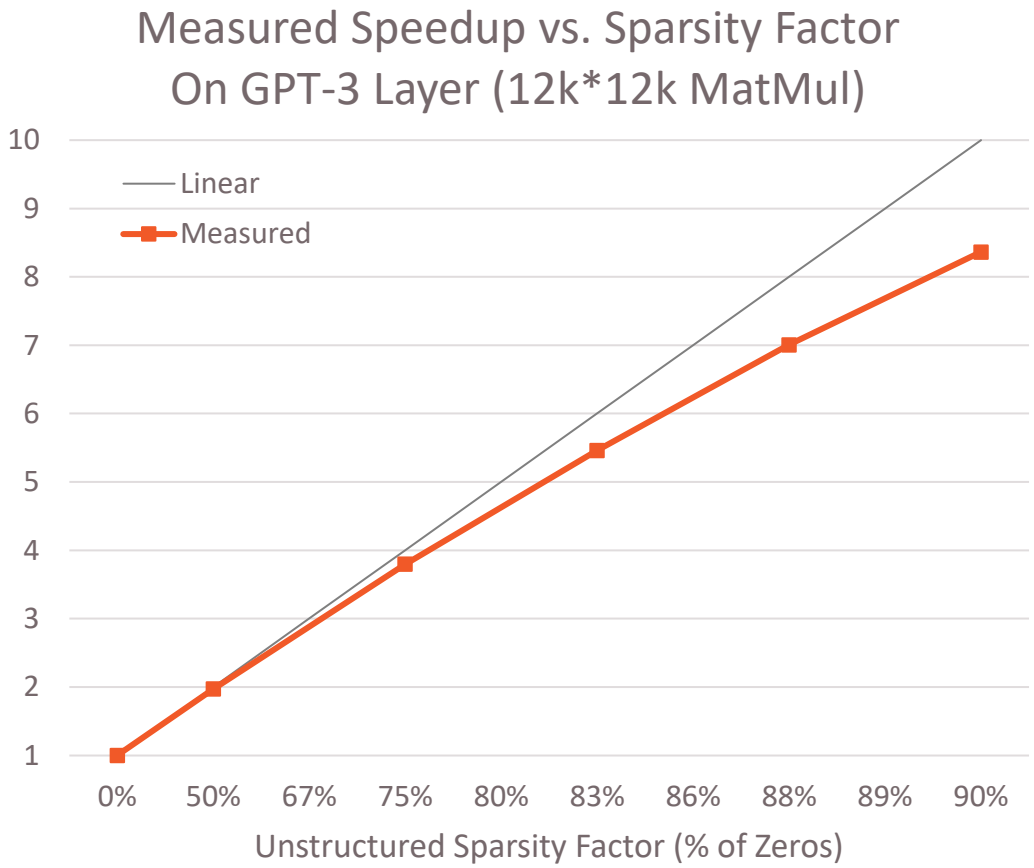


Demonstrated Unstructured Sparsity Speedup

Sparsity reduces time-to-accuracy

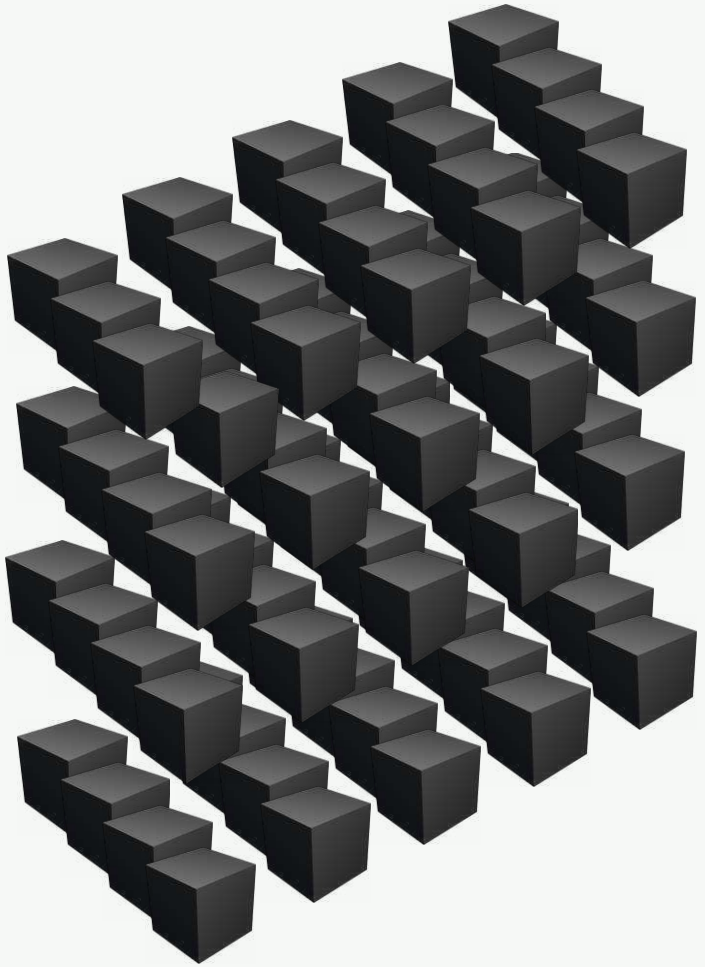
- WSE runs AXPY at full performance
- Limited only by low fixed overheads
  - Minimized by high bandwidth interconnect
  - Reduced as networks grow larger
- Accelerates all unstructured sparsity
  - Fully dynamic and fine-grained
  - Even fully random patterns

Near-linear sparsity acceleration

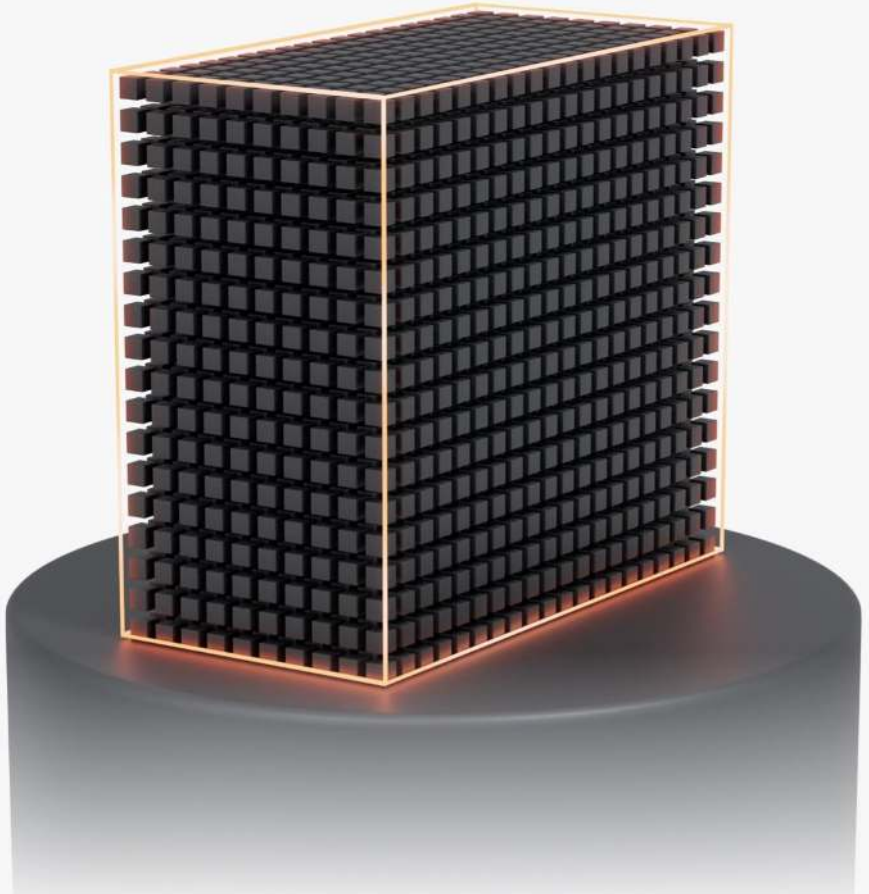


Extreme Ease

Distributed Training Today is Extremely Complicated



CS-2
Cluster-Scale Performance, Ease of a Single Node



Weight Streaming Software
Extends Programming Simplicity to Cluster-Scale

Workload maps to **multiple CS-2s the same way as for one**:

- 40 GB SRAM fits enormous layers of up to 100k hidden dim
- No complexity of partitioning individual layers or running model-parallel

Single execution model that extends to extreme-scale clusters

1. Compile the neural network mapping for a CS-2
2. Load the same mapping onto each CS-2
3. Go to town!

Running a model on a multi-CS-2 cluster **is the same** as for a single-CS-2

Just specify the number of CS-2s you want to use

120T parameter capacity on a CS-2 system

163M cores across up to 192 CS-2 systems

10x acceleration of unstructured sparsity

Push-button scaling ease

Extreme-scale models

Extreme acceleration

Extreme smarts

Extreme ease

Disaggregated scalable architecture

Imagine…

Training GPT-3 in a day

Training a 1T parameter model over a long weekend

Thank You

info@cerebras.net