

# Cerebras Systems

**The Power of Wafer-Scale Systems**



# Cerebras Systems

Building a new class of computer system for the future of AI and HPC  
A full AI acceleration solution: chip, system, software, ML



Founded in 2016

**400+ Engineers**

**Offices**

Silicon Valley | Berlin | Toronto | Bangalore

**Customers**

North America | Asia | Europe

# Our Partners and Customers

## Large Enterprise, HPC, Government

- G42, Mayo Clinic, GlaxoSmithKline, TotalEnergies, AstraZeneca, Bayer, Genentech, Tokyo Electron Devices...
- ANL, LLNL, NREL, PSC, NCSA, EPCC, Leibniz Supercomputing Centre...
- Security, e.g. DARPA, USAF, ARL

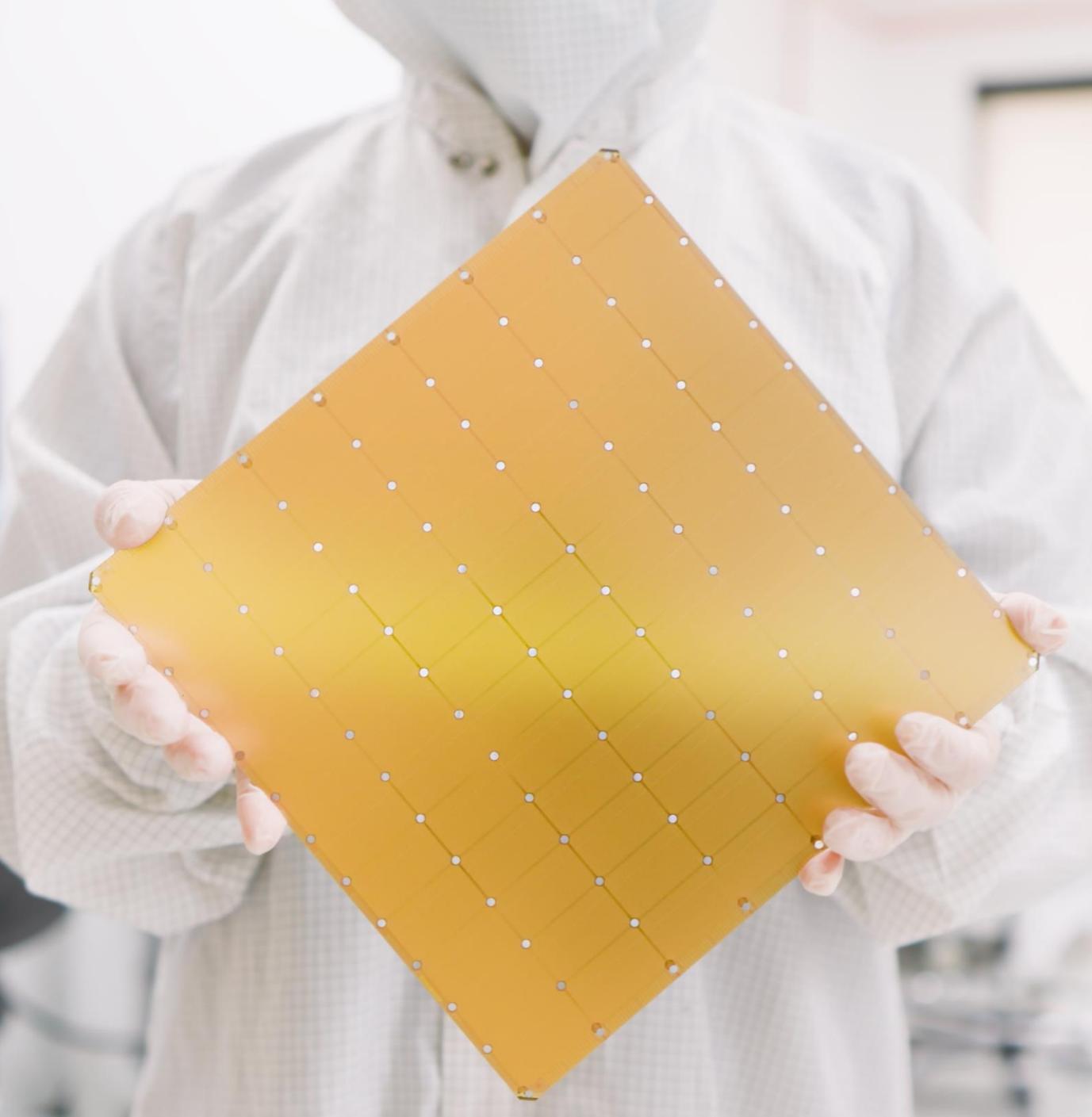


Leibniz Supercomputing Centre  
of the Bavarian Academy of Sciences and Humanities



TOKYO ELECTRON





# Cerebras Wafer-Scale Engine 3

The fastest AI chip on earth **again**

125 PFLOP/s of sparse AI compute

46,225 mm<sup>2</sup> silicon

4 trillion transistors

5nm TSMC process

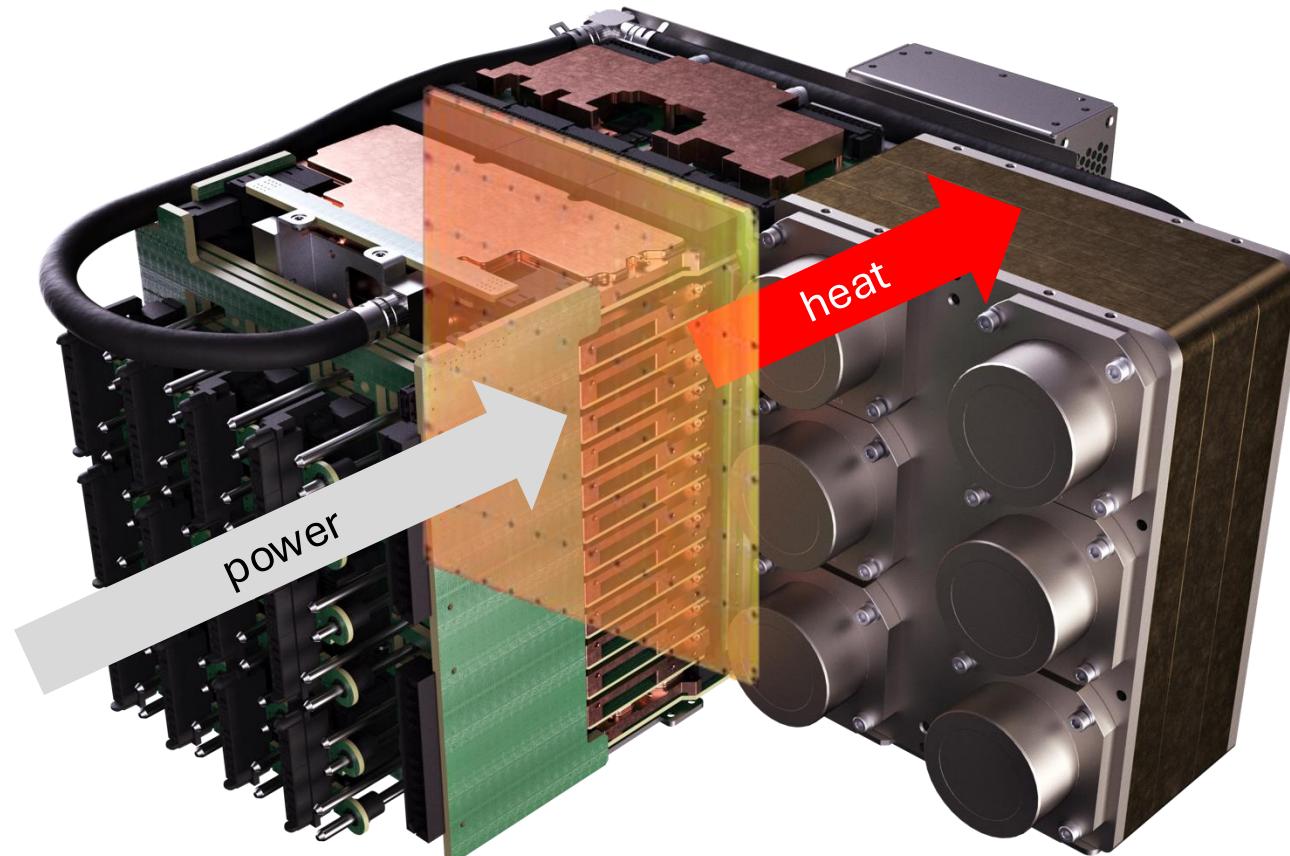
900,000 cores optimized for sparse linear algebra

44 GB of on-chip memory

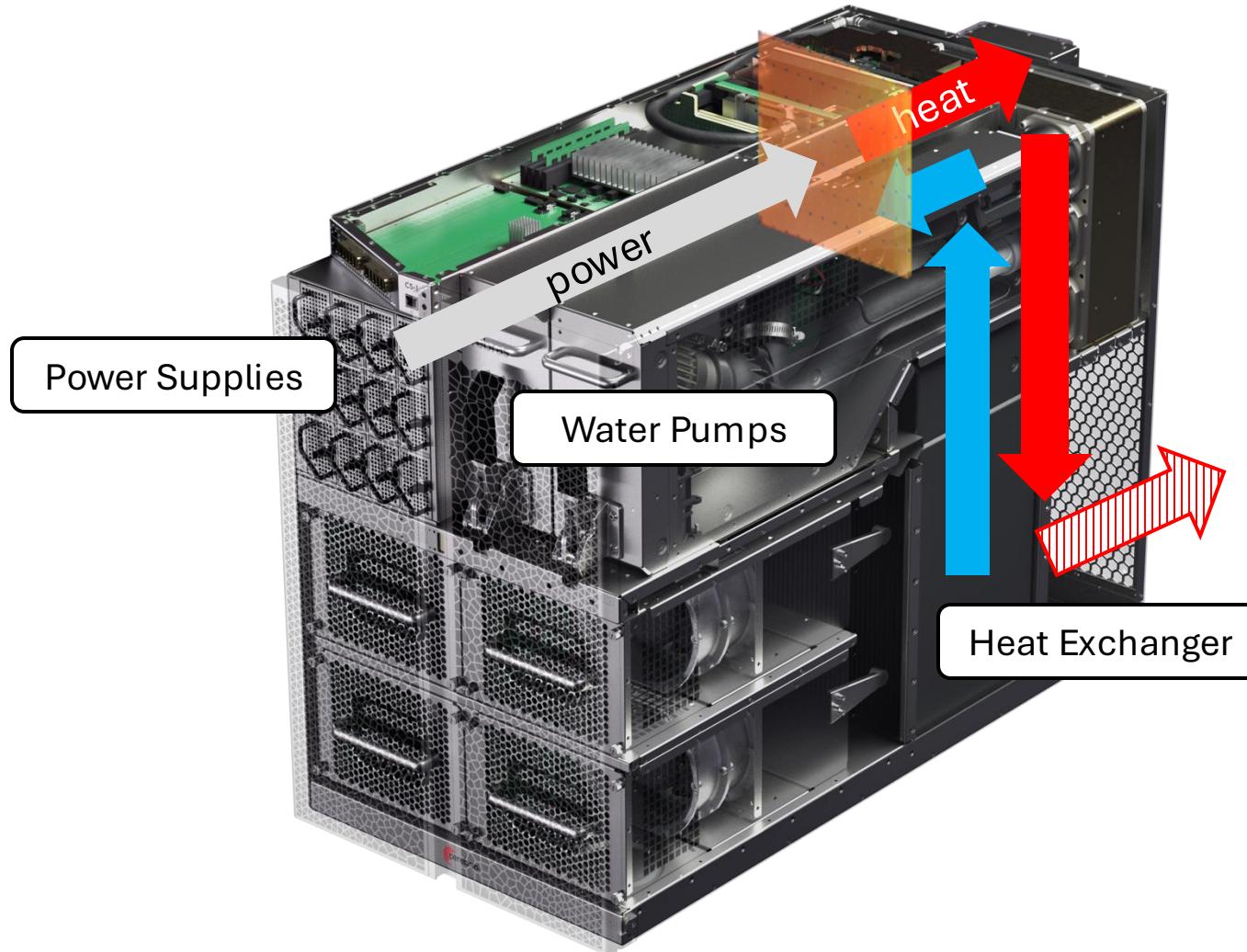
25 PB/s memory bandwidth

245 Pb/s fabric bandwidth

# The Cerebras System: Engine Block



# The Cerebras System: Chassis











Santa Clara, California  
8 Exaflops



Dallas, Texas  
20 Exaflops



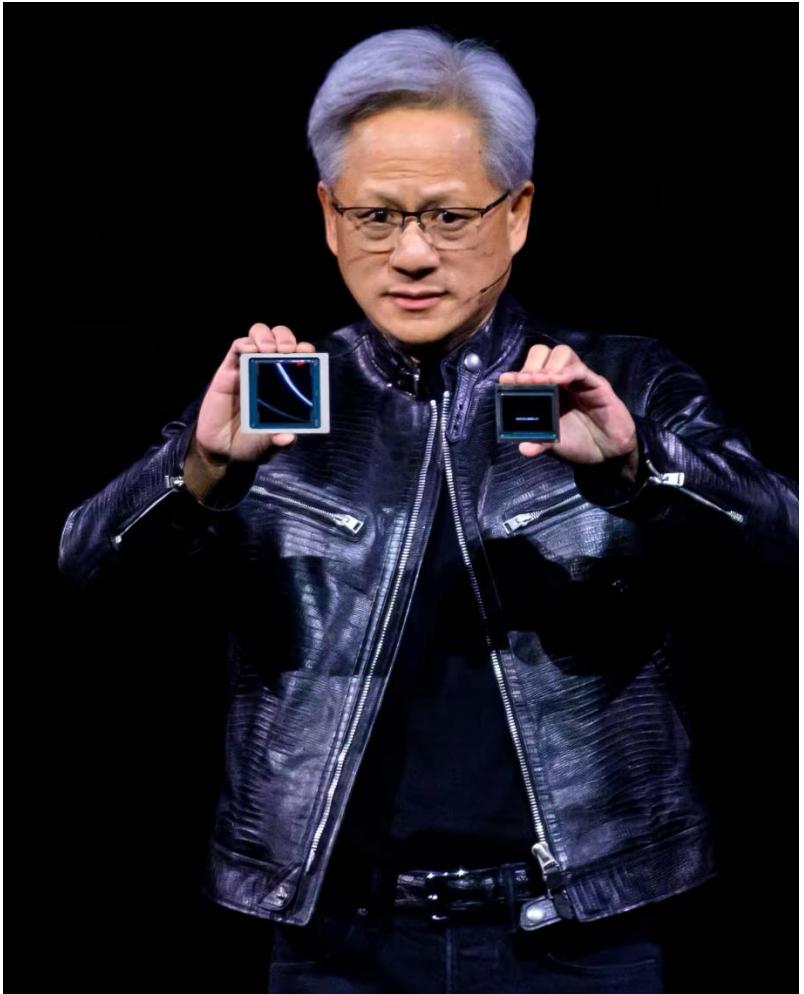
Minneapolis, MN

64 ExaFLOPs

# Inference Deep Dive

Bigger chips are faster and more efficient

---



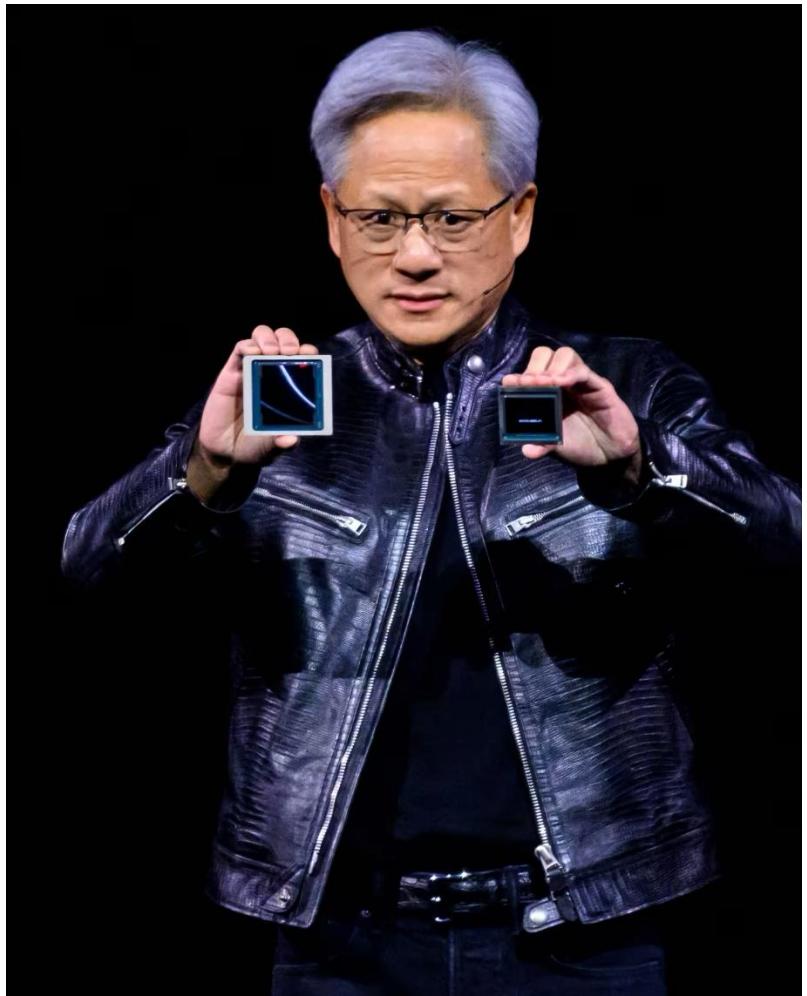
Nvidia future B200  
**2x larger than H100**  
280 Billion Transistors

*“We need **bigger GPUs.**”*

Jensen Huang  
Nvidia CEO  
Mar 19, 2024

Bigger chips are faster and more efficient

---



Nvidia future B200  
**2x larger than H100**  
280 Billion Transistors

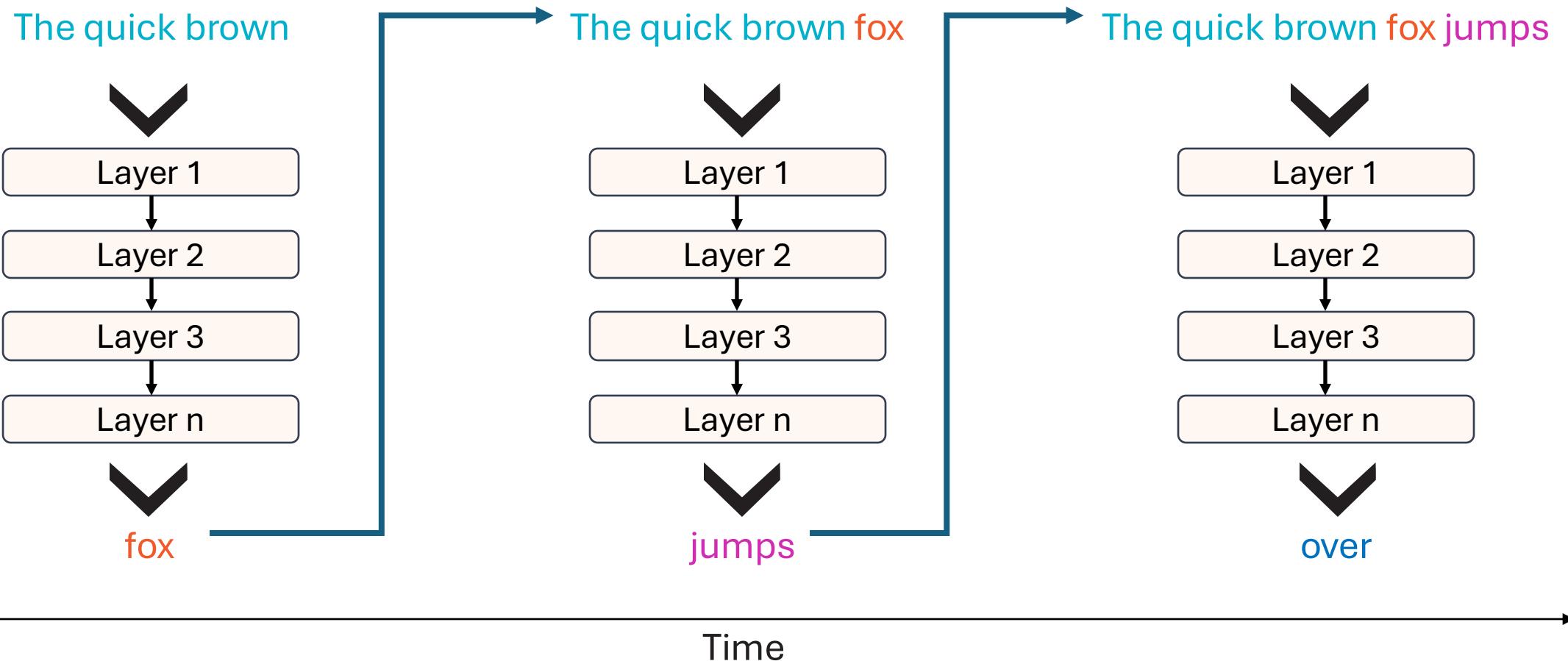


Cerebras current WSE-3  
**57x larger than H100**  
4 Trillion Transistors

# Memory Bandwidth

# GenAI inference is a **memory bandwidth** problem

- Generating 1000 tokens takes 1000 serial passes through the model
- Each pass requires reading all model parameters from memory
- Low memory bandwidth is the bottleneck for generation performance



# Why Wafer Scale Matters

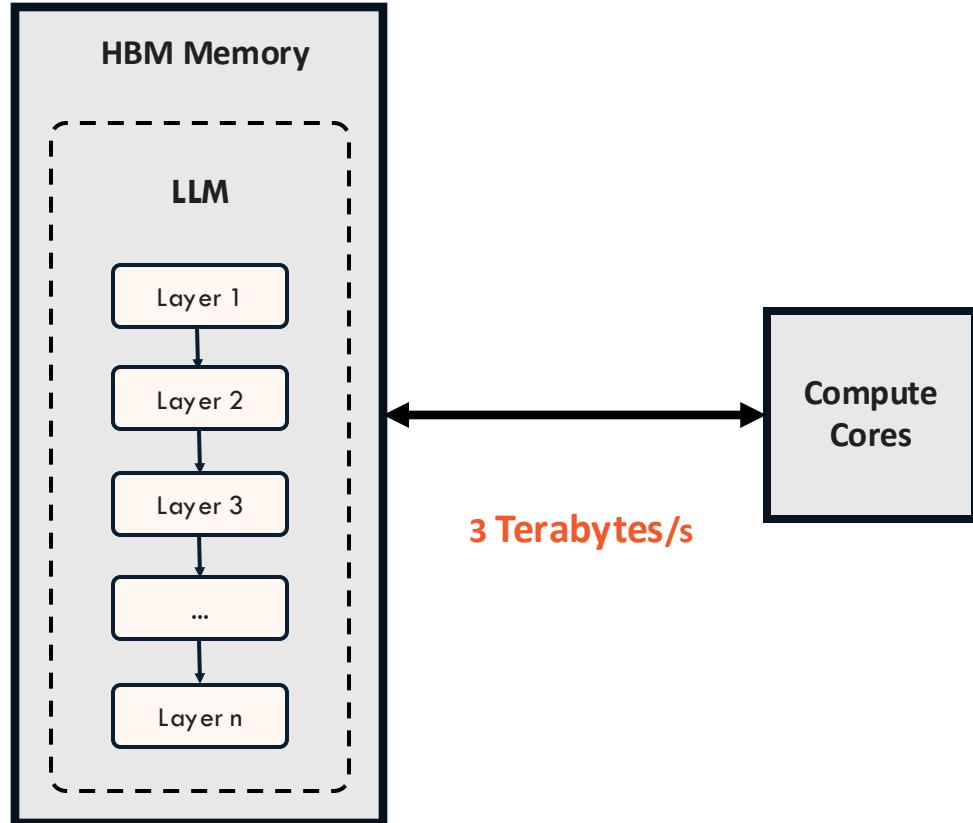
	Cerebras WSE-3	Nvidia H100	Cerebras Advantage
Chip size	46,225 mm <sup>2</sup>	814 mm <sup>2</sup>	<b>57x</b>
Cores	900,000	16,896 FP32 + 528 Tensor	<b>52x</b>
On-chip memory	44 Gigabytes	0.05 Gigabytes	<b>880x</b>
Memory bandwidth	25 Petabytes/s	0.003 Petabytes/s	<b>7,000x</b>

Completely removes the memory bandwidth bottleneck

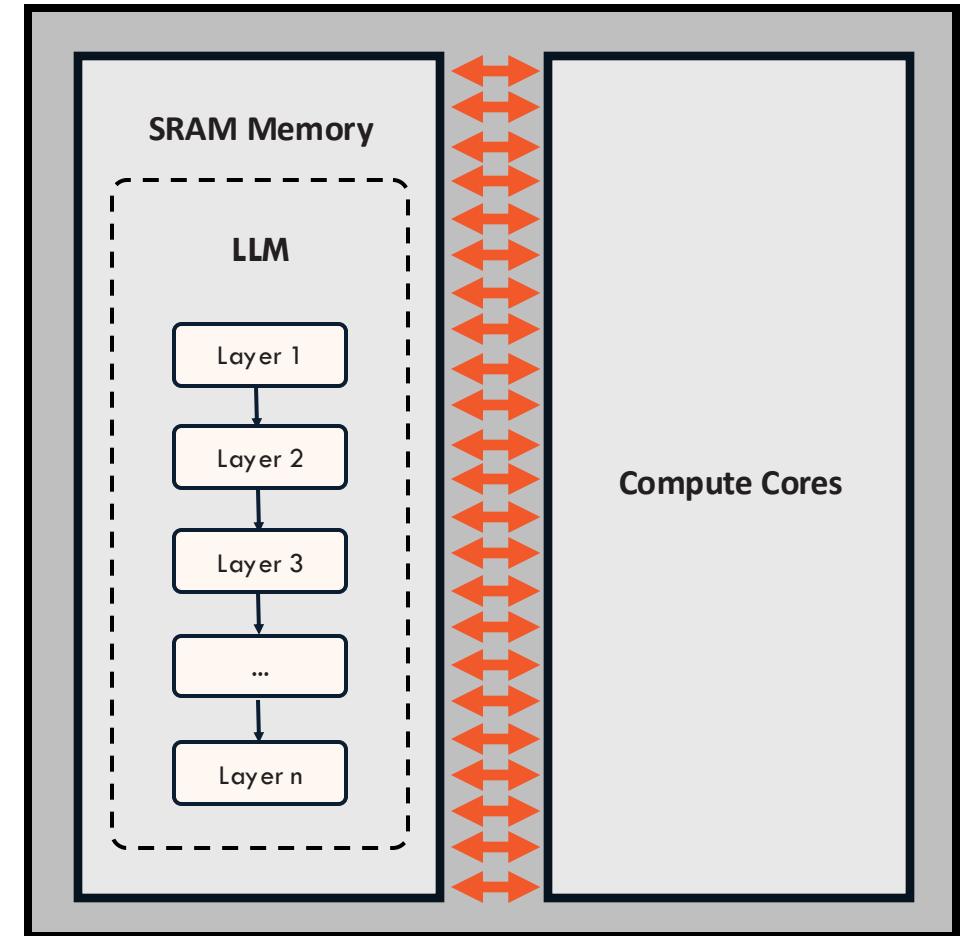
# Wafer Scale, SRAM based, Compute-in-Memory

## 7,000x more memory bandwidth than the GPU

Nvidia H100



Cerebras Wafer Scale Engine 3

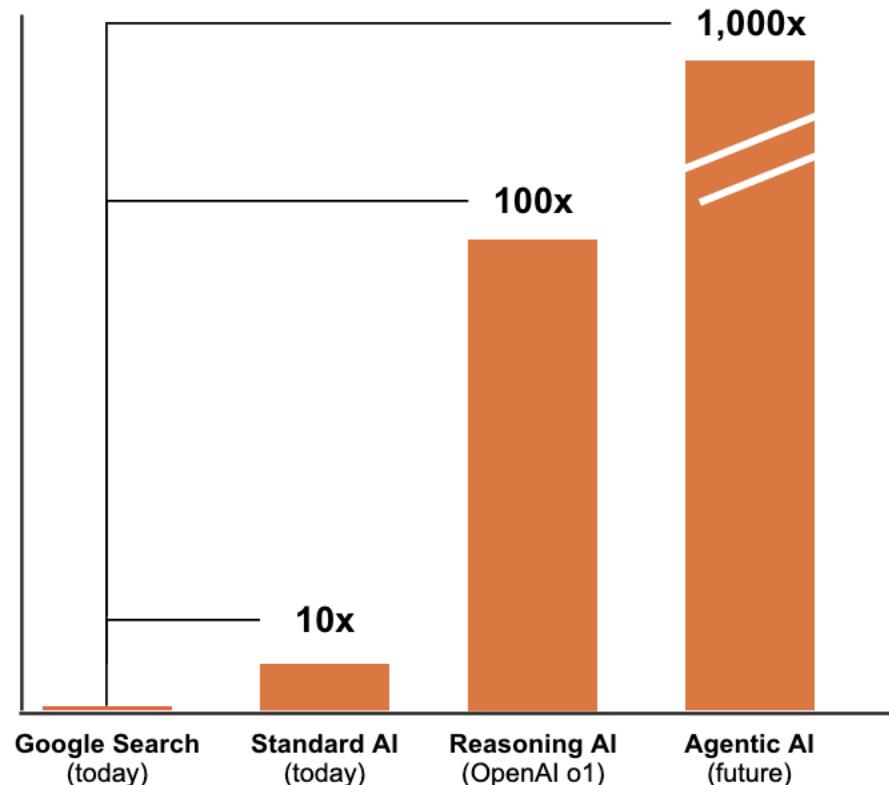


21 Petabytes/s

**Fast Inference unlocks  
new capabilities**

# 20x speed enables high user engagement and greater model capability

Illustrative time required per query  
Of different model types



## AI application latency limits mass adoption

Google shows that latency destroys user engagement:

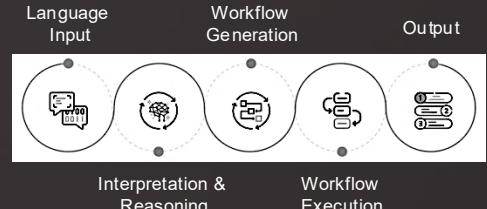
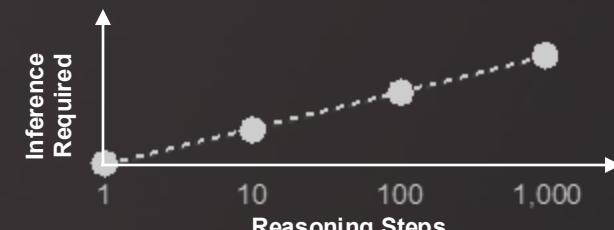
- "If results are slowed by even a fraction of a second, people search less."
- "4/5 internet users will click away if a video stalls while loading".
- Search takes milliseconds, but LLMs can take seconds to tens of minutes to respond

## Reasoning & Agentic flows require even more speed

- Trading inference calls for greater chain-of-thought reasoning
- OpenAI o1 takes minutes to respond today

↑ Speed = ↑ Engagement & Capability = ↑ ROI

# Fast inference = instant responses, smarter agents, deeper reasoning.

Application	GPUs	Illustrative examples of 70x speedup	Cerebras
Generative text & search <i>Inference queries take <b>10x longer</b> than traditional search queries</i>	Q: "Where to visit in Italy?" 5 seconds	<b>0.07 seconds</b>	
Generative voice & video <i>Require &lt;100ms latency</i>	   700 milliseconds	<b>10 milliseconds</b>	
Speed is critical in existing  and emerging  applications. Google has shown that search results need to return in milliseconds to capitalize on user engagement.			
AI agents <i>Can think <b>10+ minutes per query</b></i>	 70 minutes	<b>1 minute</b>	
Reasoning models <i>Those like OpenAI o1 conduct <b>100x more inference</b> per query</i>	 10 reasoning steps	<b>700 reasoning steps</b>	

# Inference Offering Summary

# Cerebras Ecosystem Availability

## AI Applications



## Ecosystem Integrations



## Inference API



## Models



## Hardware

Cerebras CS-3

# Experience Cerebras Inference speeds at inference.cerebras.ai

The screenshot shows the cerebras inference AI interface. At the top, there's a navigation bar with links for "Get API Key", "Learn More", "Chat With Us", and "Policies". To the right of the navigation bar are buttons for "Clear" and "Llama3.1-70B" with a dropdown arrow. A user profile icon is also present.

In the main content area, there's a section titled "ASSISTANT" on the left and a performance metric "566 T/s" with a lightning bolt icon and a question mark icon on the right. Below this, a text block states: "Ridiculously fast LLM (Large Language Model) inference would unlock a wide range of exciting possibilities. Here are some potential applications:"

- 1. Real-time Conversational AI:** Fast inference would enable conversational AI systems to respond quickly and naturally, making them feel more human-like. This could revolutionize customer service, virtual assistants, and chatbots.
- 2. Instant Language Translation:** Rapid inference would allow for real-time language translation, breaking down language barriers and enabling more effective global communication.
- 3. Efficient Content Generation:** Fast LLM inference would accelerate content generation, such as automated writing, summarization, and text analysis. This could benefit industries like publishing, marketing, and education.
- 4. Personalized Recommendations:** Quick inference would enable personalized recommendation systems to provide instant, context-aware suggestions for products, services, or content.
- 5. Enhanced Accessibility:** Fast LLM inference could improve accessibility features like text-to-speech, speech-to-text, and language translation for people with disabilities.
- 6. Real-time Sentiment Analysis:** Rapid inference would allow for instant sentiment analysis, enabling businesses to respond quickly to customer feedback and sentiment shifts.
- 7. Intelligent Tutoring Systems:** Fast LLM inference could power intelligent tutoring systems that provide personalized, real-time feedback and guidance to students.
- 8. Efficient Data Analysis:** Quick inference would accelerate data analysis, enabling faster insights and decision-making in fields like finance, healthcare, and scientific research.
- 9. Improved Search Engines:** Fast LLM inference could enhance search engines with more accurate, context-aware results, and instant answers to complex queries.
- 10. New Forms of Human-Computer Interaction:** Ridiculously fast LLM inference could enable new forms of human-computer interaction, such as voice-controlled interfaces, augmented reality experiences, and more.

Below this list, a note says: "These are just a few examples of the many exciting possibilities that ridiculously fast LLM inference could unlock. The actual applications will depend on the creativity and innovation of developers, researchers, and industries that leverage".

At the bottom, there's a large input field with the placeholder "Ask anything..." and a blue send button with a white arrow. Below the input field is a grey bar with the text "Try Voice Mode" and a microphone icon. At the very bottom, a small note reads "Meta-based chatbot. Verify results for accuracy".

# Experiment with models in our Developer Playground

The screenshot shows the Cerebras Developer Playground interface. On the left is a sidebar with links like 'Playground' (which is selected), 'API Keys', 'Usage & Limits', 'Pricing', 'Settings', 'Documentation', 'Chat Demo', 'Why Cerebras?', and 'Say Hello!'. Below the sidebar are social media icons for LinkedIn and Twitter.

The main area has a 'View code' button and a 'Clear' button at the top. A dropdown menu shows 'Llama3.1-70B' and a user profile icon. To the right are several configuration sliders: 'Temperature' (0.5), 'Max tokens' (512), 'Top P' (0.5), 'Seed' (1), and 'Stream' (on). There's also a 'JSON' toggle and a 'Stop sequence' input field.

The central conversation window shows a 'SYSTEM' prompt: 'Please answer as if you are an expert at model inferencing'. Below it, a 'USER' message asks: 'Why is high memory bandwidth so important to fast GenAI model inferencing?'. The 'ASSISTANT' response is: 'As an expert in model inferencing, I can tell you that high memory bandwidth is crucial for fast GenAI (Generative Artificial Intelligence) model inferencing because of the way these models are designed and the computations they perform.' It includes a bolded '452 T/s' with a lightning bolt icon. The assistant then explains: 'GenAI models, such as transformers and LSTMs, rely heavily on matrix multiplications and tensor operations to process input data and generate output. These operations involve accessing and manipulating large amounts of data in memory, which requires high memory bandwidth to perform efficiently.' Finally, it lists reasons: 'Here are some reasons why high memory bandwidth is essential for fast GenAI model inferencing:  
1. Matrix multiplications: GenAI models perform numerous matrix multiplications, which involve accessing and multiplying large matrices. High memory bandwidth allows for faster access to these matrices, reducing the time it takes.'

At the bottom, there's a 'Message Cerebras' input field, a 'User' tab, an 'Assistant' tab, an 'Add' button, and a 'Run' button.

# Switch to Cerebras Inference in 30 seconds with OpenAI-compatible API

```
import os
from openai import OpenAI

client = OpenAI(
    api_key=os.environ.get("OPENAI_API_KEY"),
)

response = client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[...],
    temperature=1,
    max_tokens=256,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0
)
```



```
import os
from openai import OpenAI

client = OpenAI(
    base_url="https://api.cerebras.ai/v1",
    api_key=os.environ.get("CEREBRAS_API_KEY"),
)

response = client.chat.completions.create(
    model="llama3.1-70b",
    messages=[...],
    temperature=1,
    max_tokens=256,
    top_p=1,
    frequency_penalty=0,
    presence_penalty=0
)
```

Features  
**Function calling**  
**Json output**

Coming soon  
**Structured output**

# Cerebras Inference Solutions



## Pay-As-You-Go

- Serverless API endpoint charged by millions of tokens
- Easily scale up or down
- No SLA. Rates and token limits subject to change
- Ideal for early-stage exploration and experimentation



## Provisioned Throughput

- Serverless API endpoint with guaranteed performance
- Fine-tuned model support
- Fixed monthly or yearly charge
- Minimum commitment
- Ideal for startups with consistent, predictable workloads



## Private System via Cerebras Cloud

- Private deployment managed by Cerebras
- Full control over data residency
- Maintained by Cerebras
- Ideal for sustained workloads of digital-native enterprises



## Private System on Premises

- Deployment in customer's data center
- Customer fully owns infrastructure and information
- Hardware optimization for specific workloads
- Long-term cost savings
- Ideal for large enterprise, government, healthcare & secure AI solutions

*Serverless endpoint*

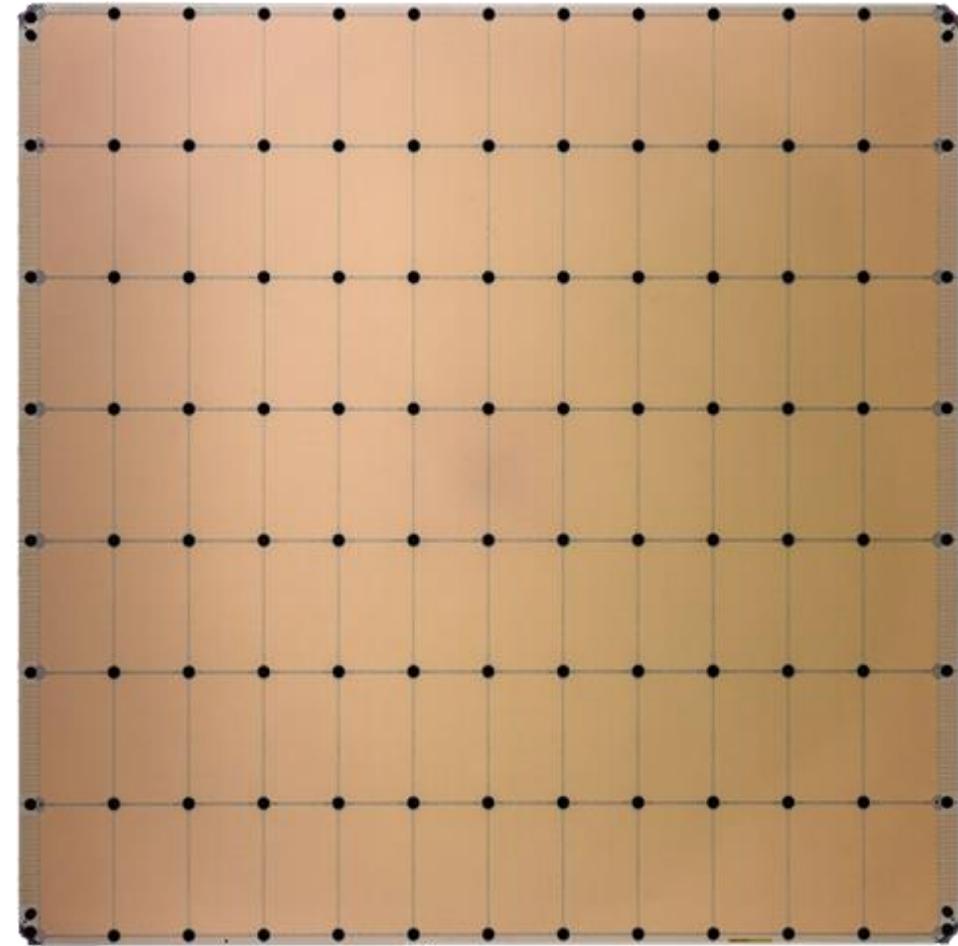
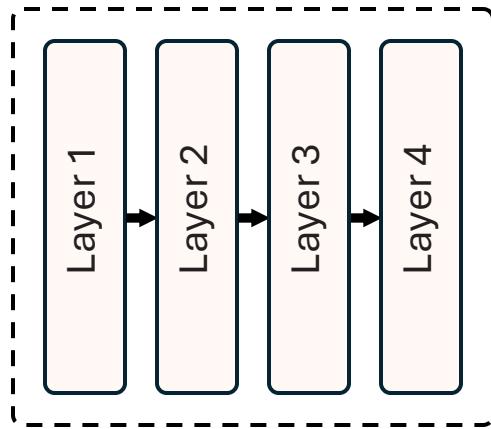
*Private deployment*

# Lowest Latency Generation

# Pipeline execution on a single chip

**Massive memory bandwidth enables  
opposite execution model**

- GPU: multiple chips to run a single layer
- Cerebras: fraction of a chip to run a single layer



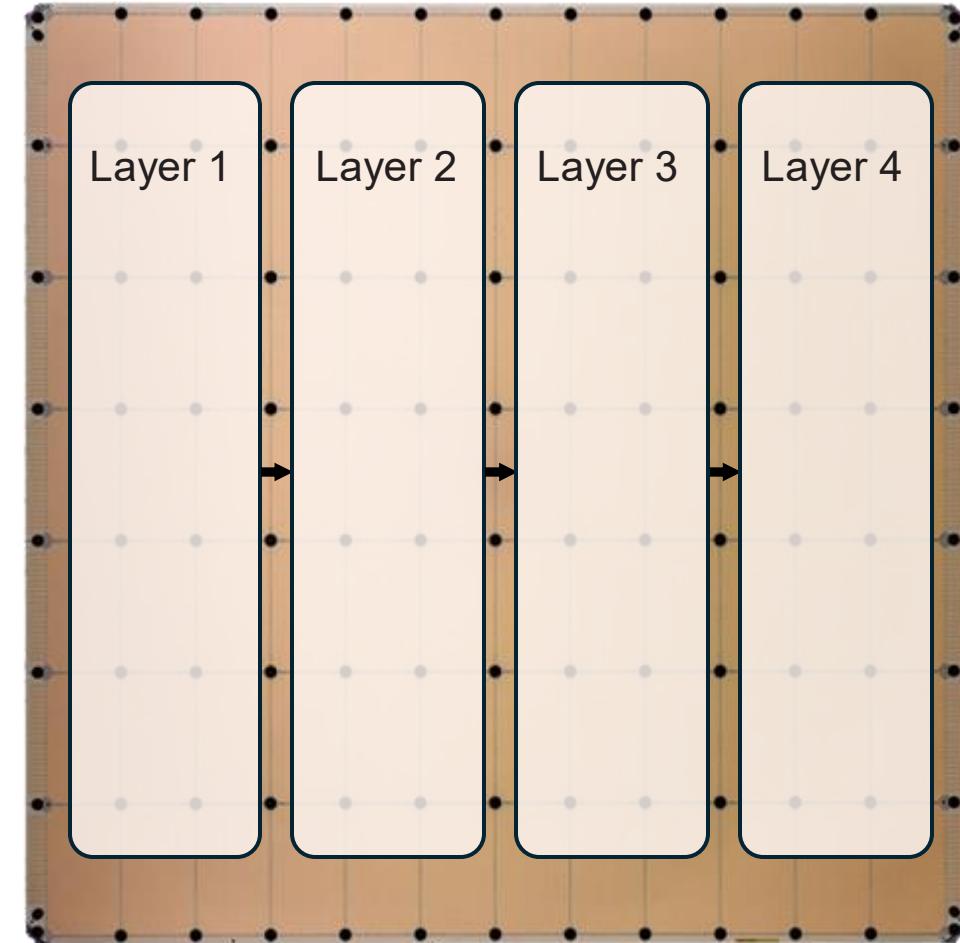
# Pipeline execution on a single chip

**Massive memory bandwidth enables opposite execution model**

- GPU: multiple chips to run a single layer
- Cerebras: fraction of a chip to run a single layer

**Pipeline mapping of model**

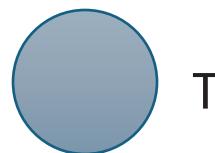
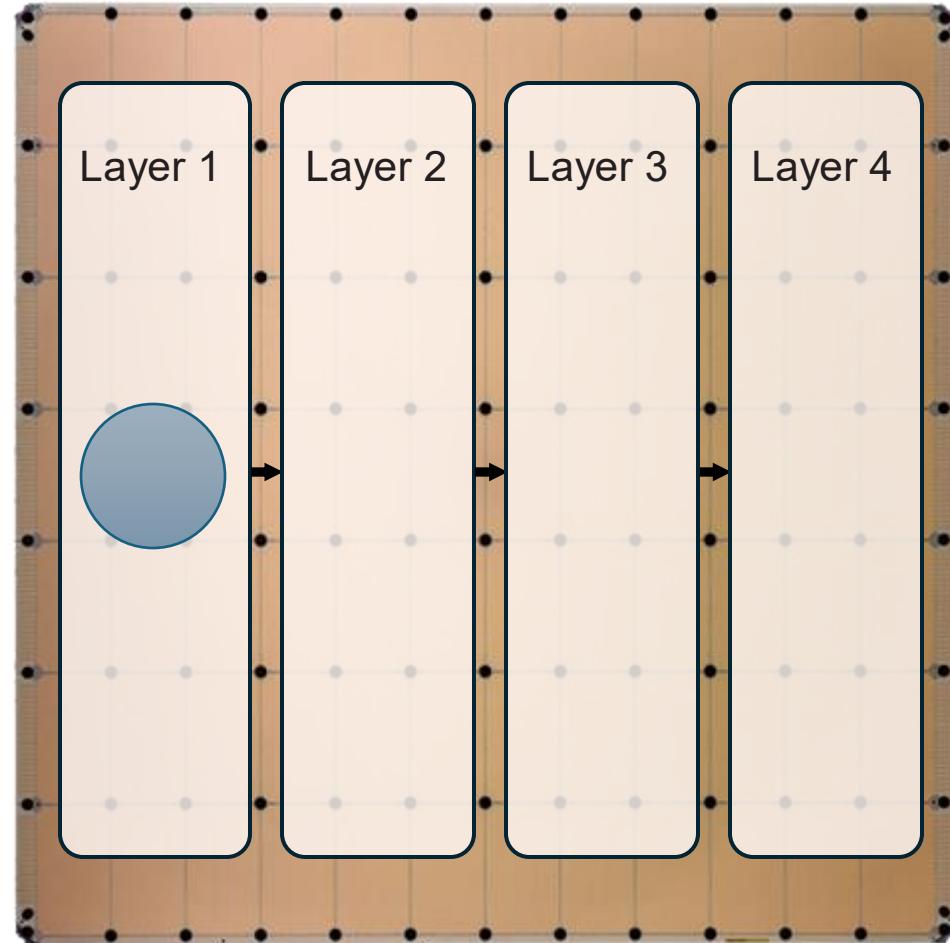
- Model layers are mapped to wafer regions
- Size of wafer region determined based on memory and compute requirements
- Model weights and KV cache stored locally in the region memory close to the compute



# Pipeline execution

## A low latency pipeline

- Each wafer region processes 1 token
  - Enough memory bandwidth to run local batch size 1
  - Memory to feed compute datapath at full speed
  - Enabling optimized performance for matrix\*vector
  - Local fabric interconnect to maintain low latency

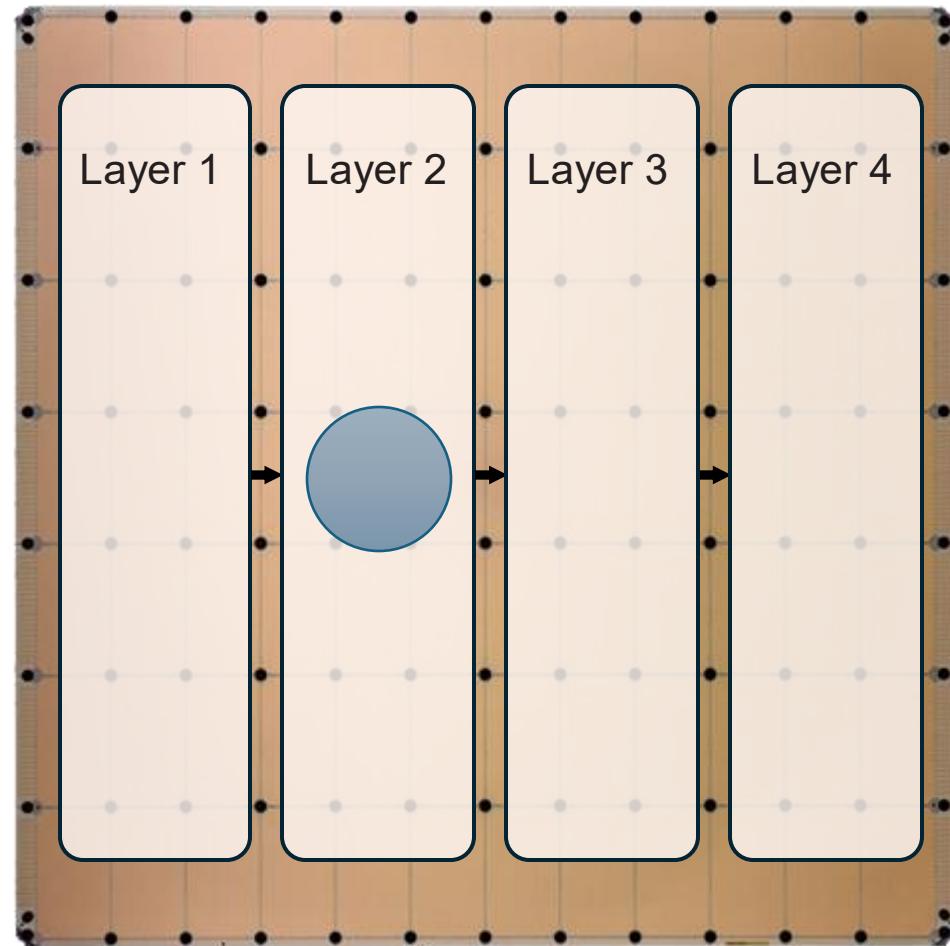


Token

# Pipeline execution

## A low latency pipeline

- Each wafer region processes 1 token
  - Enough memory bandwidth to run local batch size 1
  - Memory to feed compute datapath at full speed
  - Enabling optimized performance for matrix\*vector
  - Local fabric interconnect to maintain low latency
- Regions are placed to reduce latency
  - The next region is physically adjacent
  - Virtually no latency between pipe stages
  - Possible because it's all done on-chip

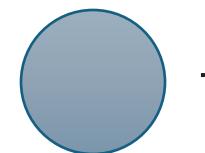
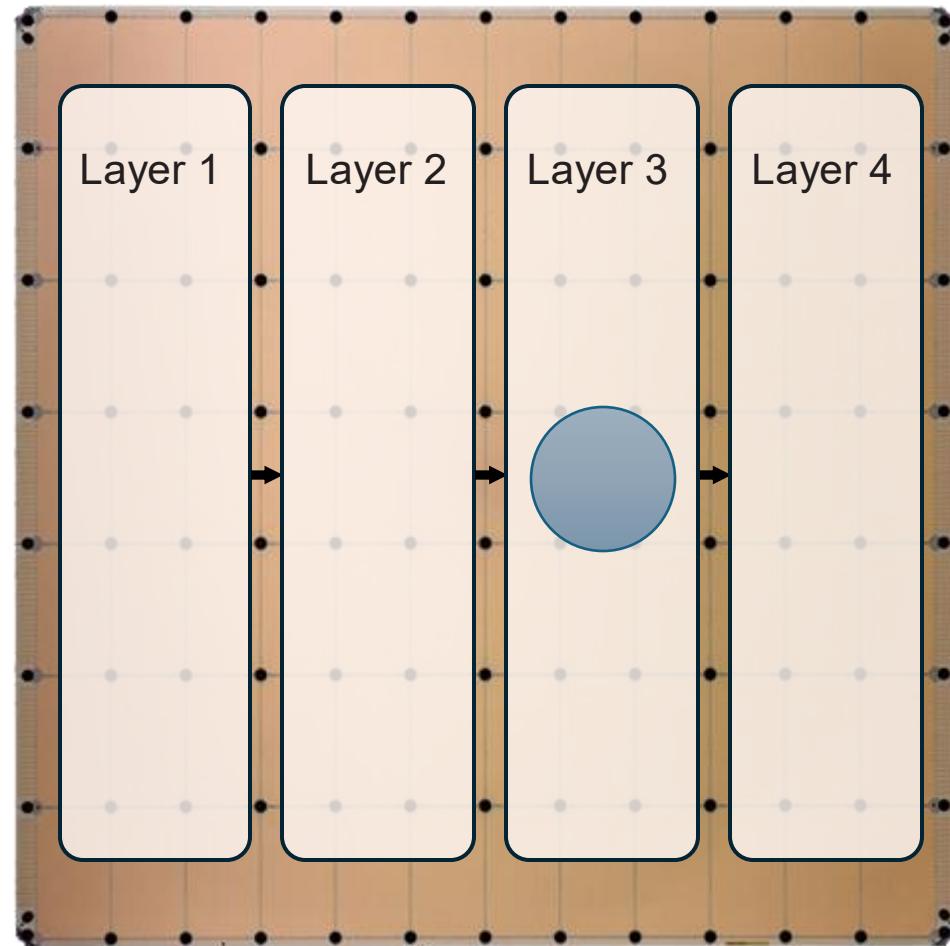


Token

# Pipeline execution

## A low latency pipeline

- Each wafer region processes 1 token
  - Enough memory bandwidth to run local batch size 1
  - Memory to feed compute datapath at full speed
  - Enabling optimized performance for matrix\*vector
  - Local fabric interconnect to maintain low latency
- Regions are placed to reduce latency
  - The next region is physically adjacent
  - Virtually no latency between pipe stages
  - Possible because it's all done on-chip

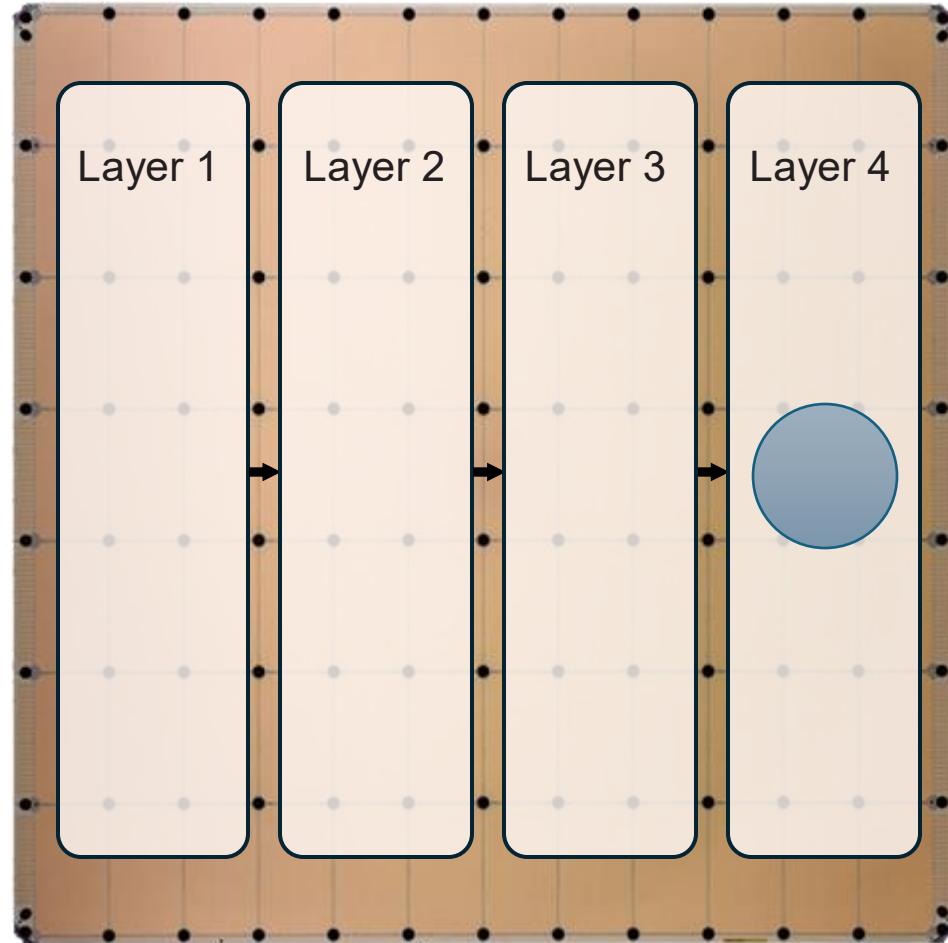


Token

# Pipeline execution

## A low latency pipeline

- Each wafer region processes 1 token
  - Enough memory bandwidth to run local batch size 1
  - Memory to feed compute datapath at full speed
  - Enabling optimized performance for matrix\*vector
  - Local fabric interconnect to maintain low latency
- Regions are placed to reduce latency
  - The next region is physically adjacent
  - Virtually no latency between pipe stages
  - Possible because it's all done on-chip



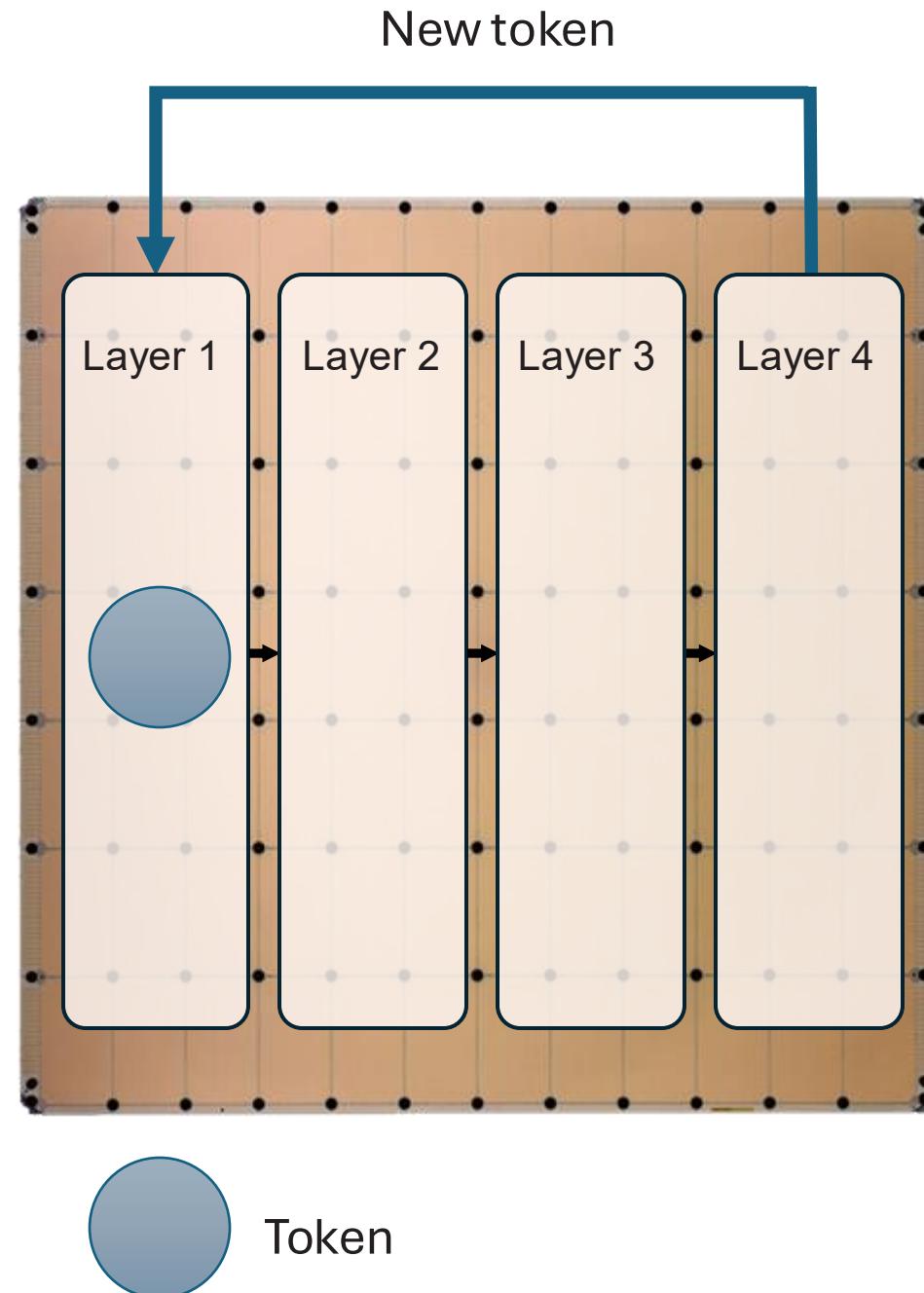
Token

# Pipeline execution

## A low latency pipeline

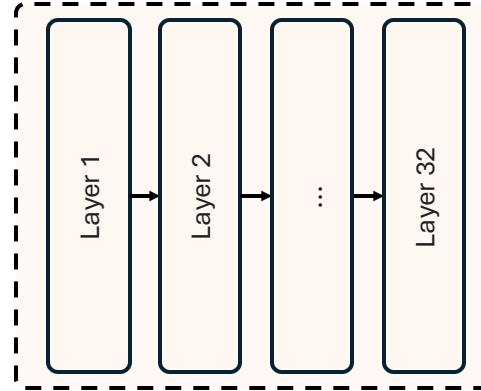
- Each wafer region processes 1 token
  - Enough memory bandwidth to run local batch size 1
  - Memory to feed compute datapath at full speed
  - Enabling optimized performance for matrix\*vector
  - Local fabric interconnect to maintain low latency
- Regions are placed to reduce latency
  - The next region is physically adjacent
  - Virtually no latency between pipe stages
  - Possible because it's all done on-chip
- Last region output cycled back to generate next token

**Pipelined execution enables super-fast token generation**



# Scalability

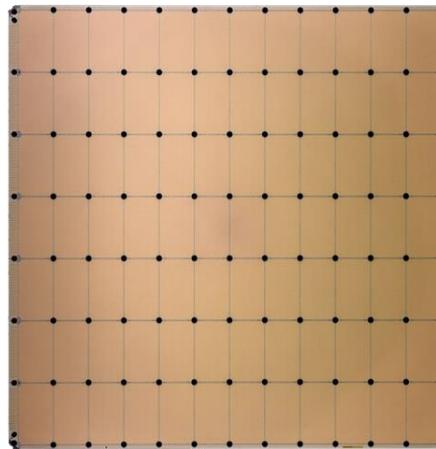
# Large models fit on our large on-chip memory



**Llama3.1-8B**

8 billion parameters

16 GB of memory (FP16)

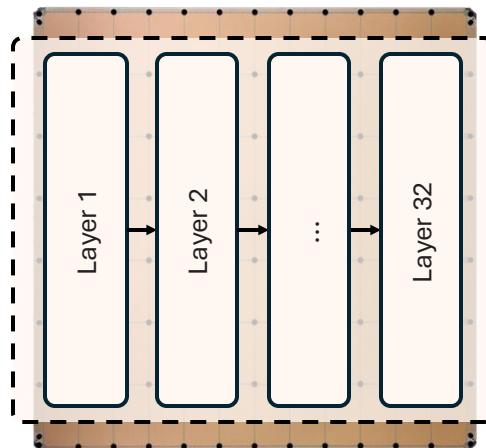


**WSE-3**

44GB of SRAM

# Large models fit on our large on-chip memory

- Models that fit entirely on the wafer are mapped directly



**Llama3.1-8B**

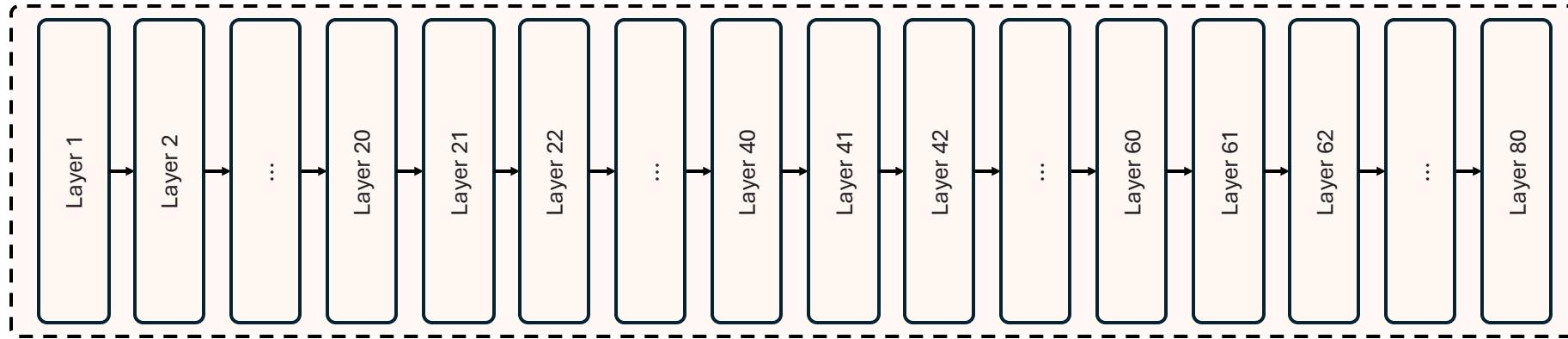
8 billion parameters

16 GB of memory (FP16)

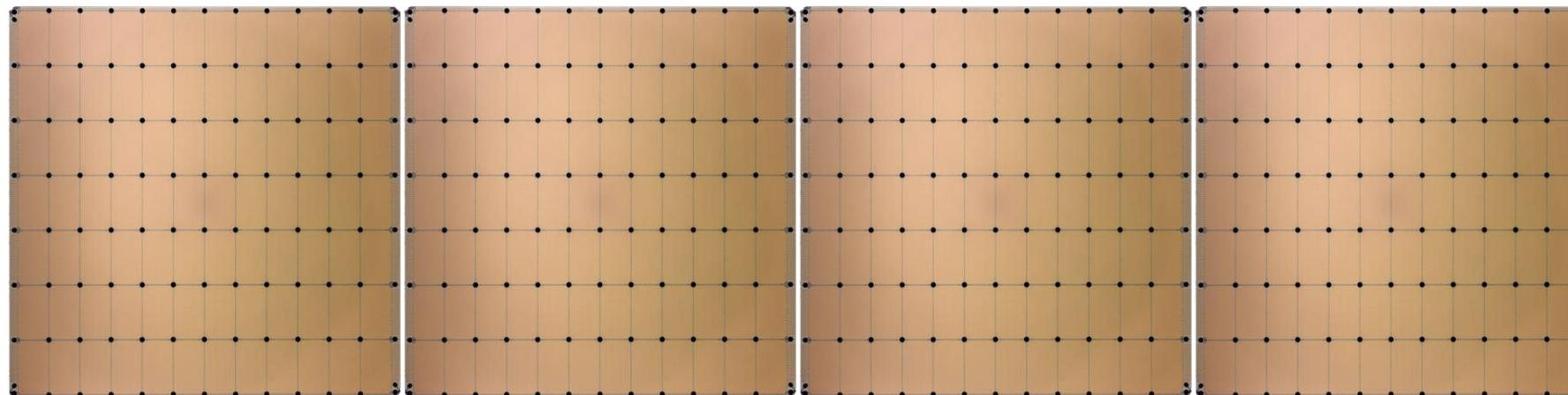
**WSE-3**

44GB of SRAM

# Naturally scales to multiple wafers



**Llama3.1-70B**  
70 billion parameters  
140 GB of memory (FP16)



**4x WSE-3**  
176GB of SRAM

# Naturally scales to multiple wafers

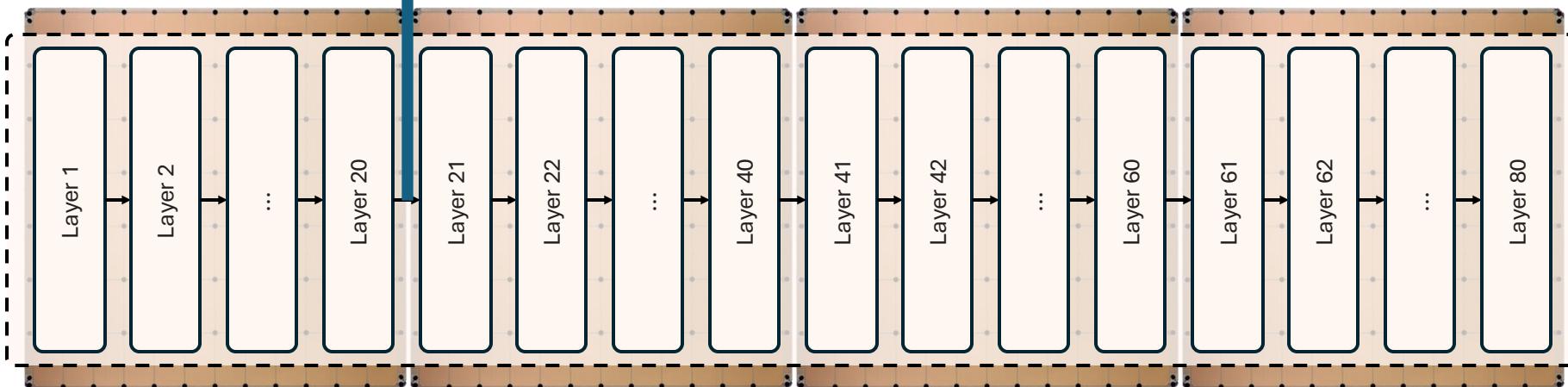
- Models that require more memory are mapped to multiple wafers
- Keep almost all high communication on wafer, using internal high bandwidth fabric
- Transfer only activations between wafers, requiring relatively lower bandwidth
- Using CS-3 low latency systems

## Latency:

CS-3 IO is <5us  
4 hops required  
<1% impact to latency

## Bandwidth:

CS-3 IO is 1.2Tbps  
100Gbps required  
<10% of available



## Llama3.1-70B

70 billion parameters  
140 GB of memory (FP16)

## 4x WSE-3

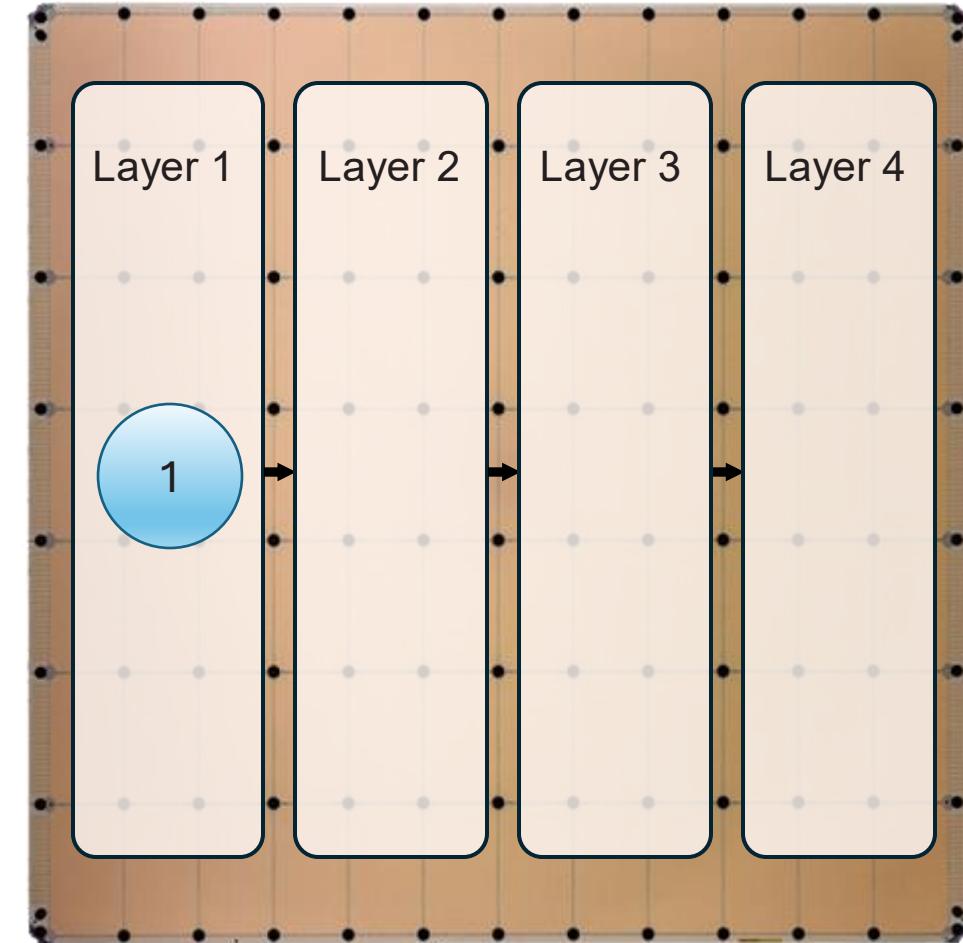
176GB of SRAM

# High Throughput

# Cerebras enables **low latency and high throughput**

More than enough memory bandwidth for single user enables high multi-user throughput

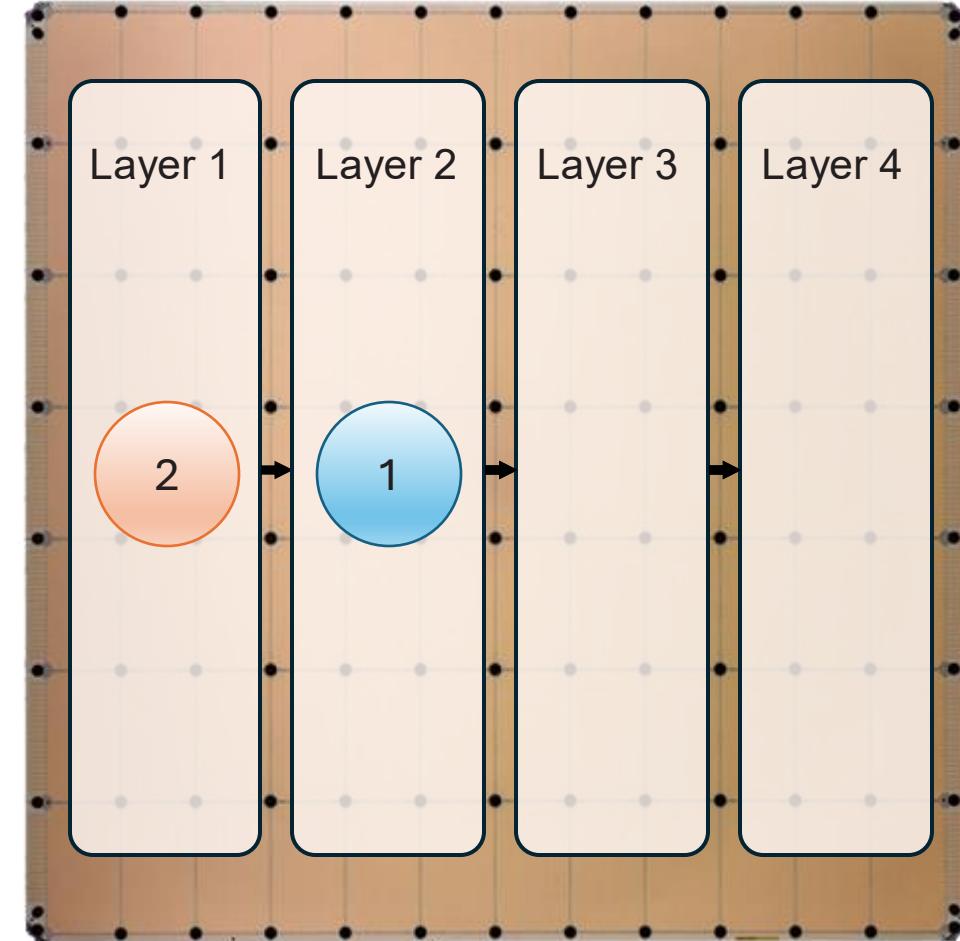
- Single user is using only a fraction of the total memory bandwidth
- We can use the extra bandwidth to support multiple users



# Cerebras enables **low latency and high throughput**

**More than enough memory bandwidth for single user enables high multi-user throughput**

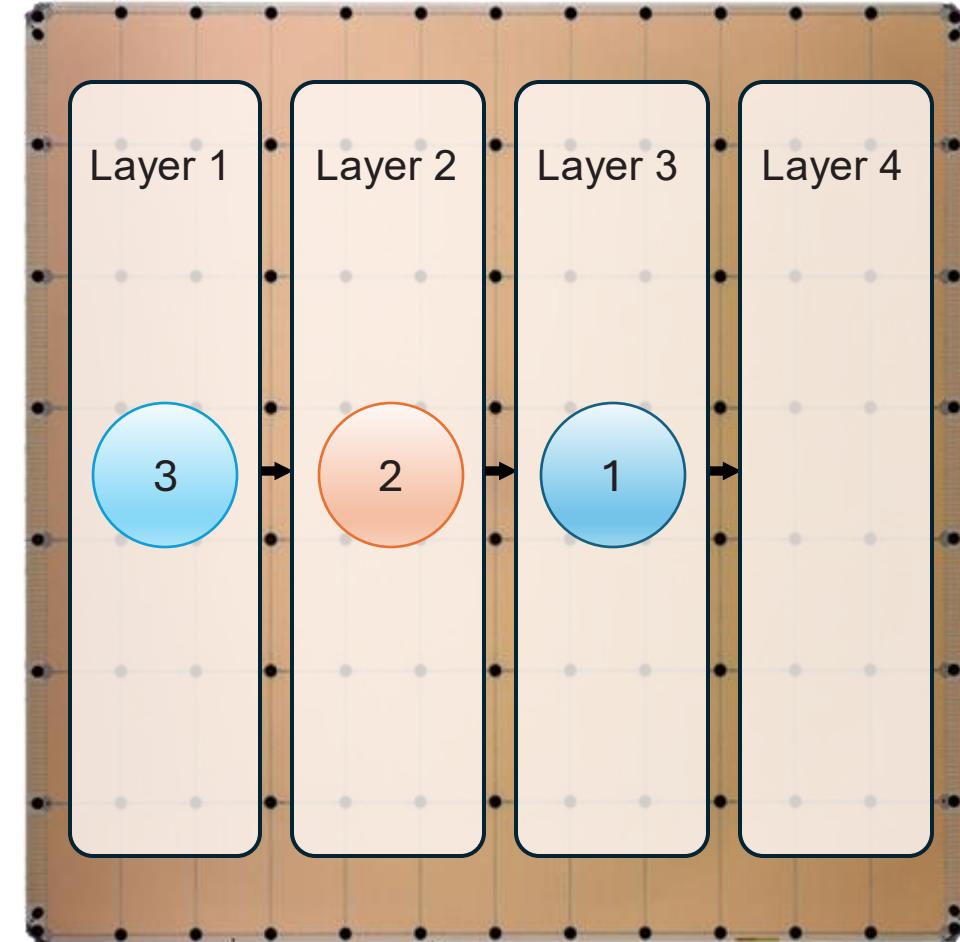
- Single user is using only a fraction of the total memory bandwidth
- We can use the extra bandwidth to support multiple users
- Additional users all run in parallel
- Each accessing the model simultaneously



# Cerebras enables **low latency and high throughput**

**More than enough memory bandwidth for single user enables high multi-user throughput**

- Single user is using only a fraction of the total memory bandwidth
- We can use the extra bandwidth to support multiple users
  - Additional users all run in parallel
  - Each accessing the model simultaneously
  - Every user gets full performance
  - All pipe stages to run at the same time

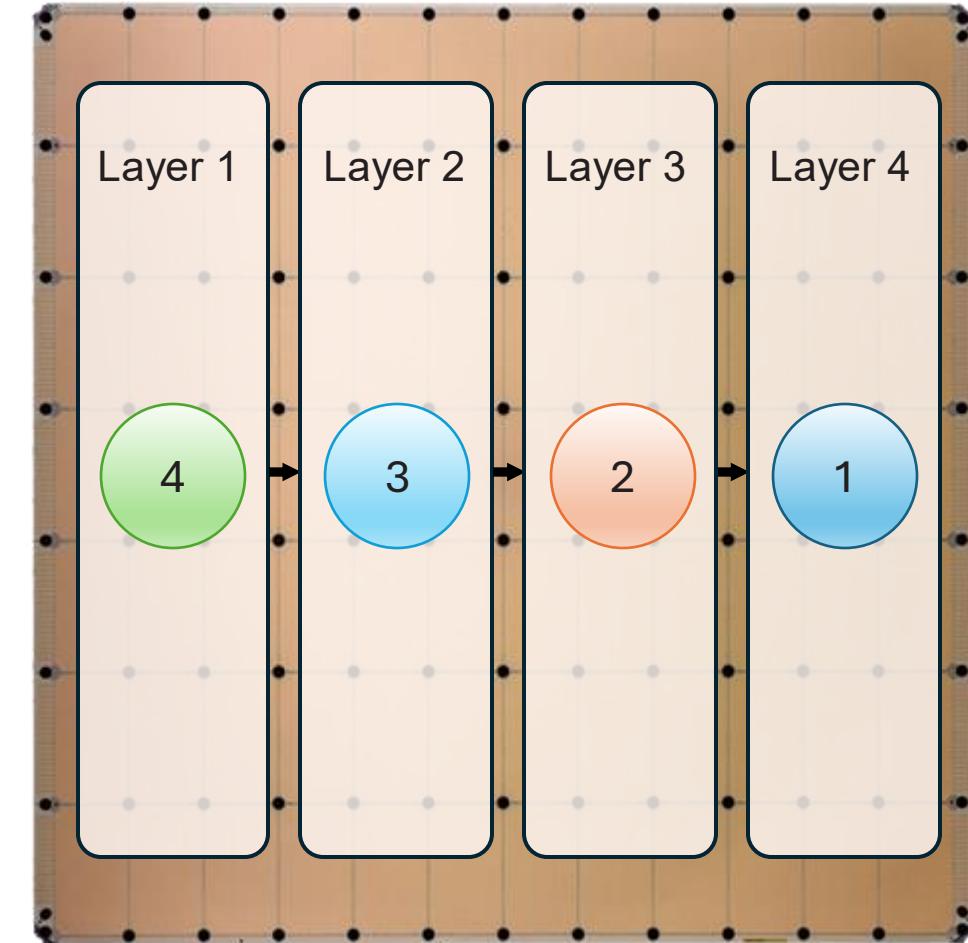


# Cerebras enables **low latency and high throughput**

**More than enough memory bandwidth for single user enables high multi-user throughput**

- Single user is using only a fraction of the total memory bandwidth
- We can use the extra bandwidth to support multiple users
  - Additional users all run in parallel
  - Each accessing the model simultaneously
  - Every user gets full performance
  - All pipe stages to run at the same time

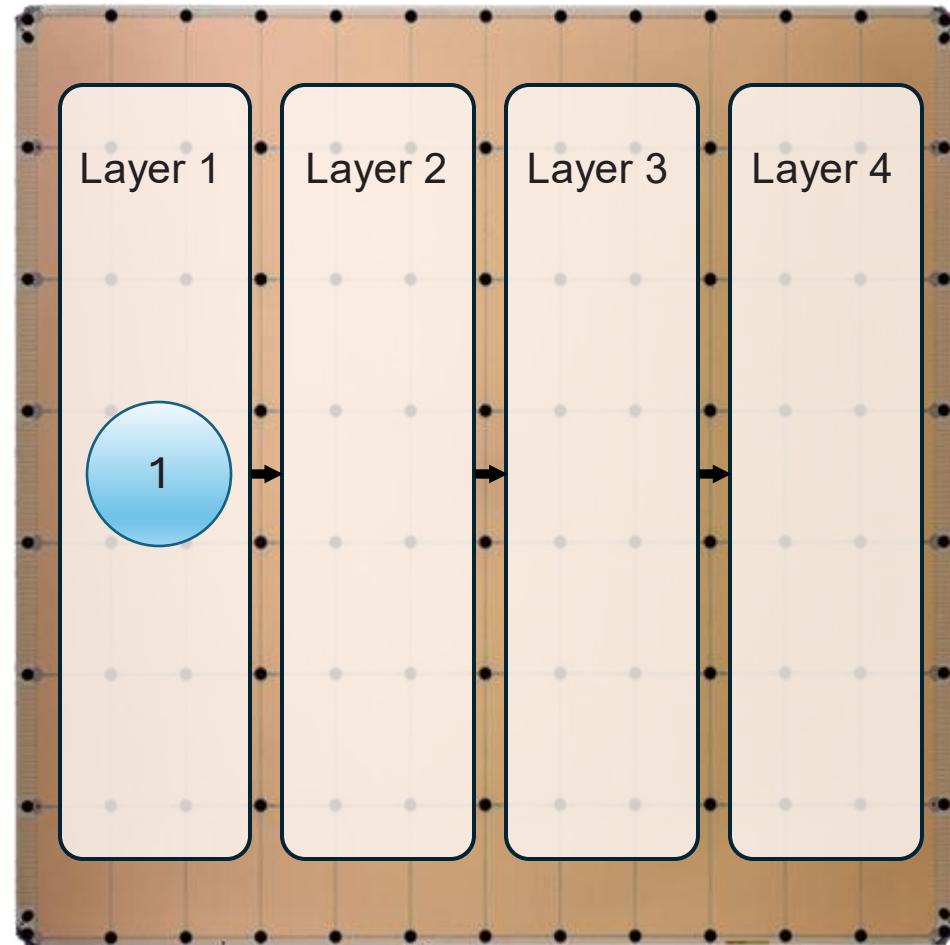
**Full pipeline model parallelism on single chip**



# High performance input processing

## Input tokens can be processed in parallel

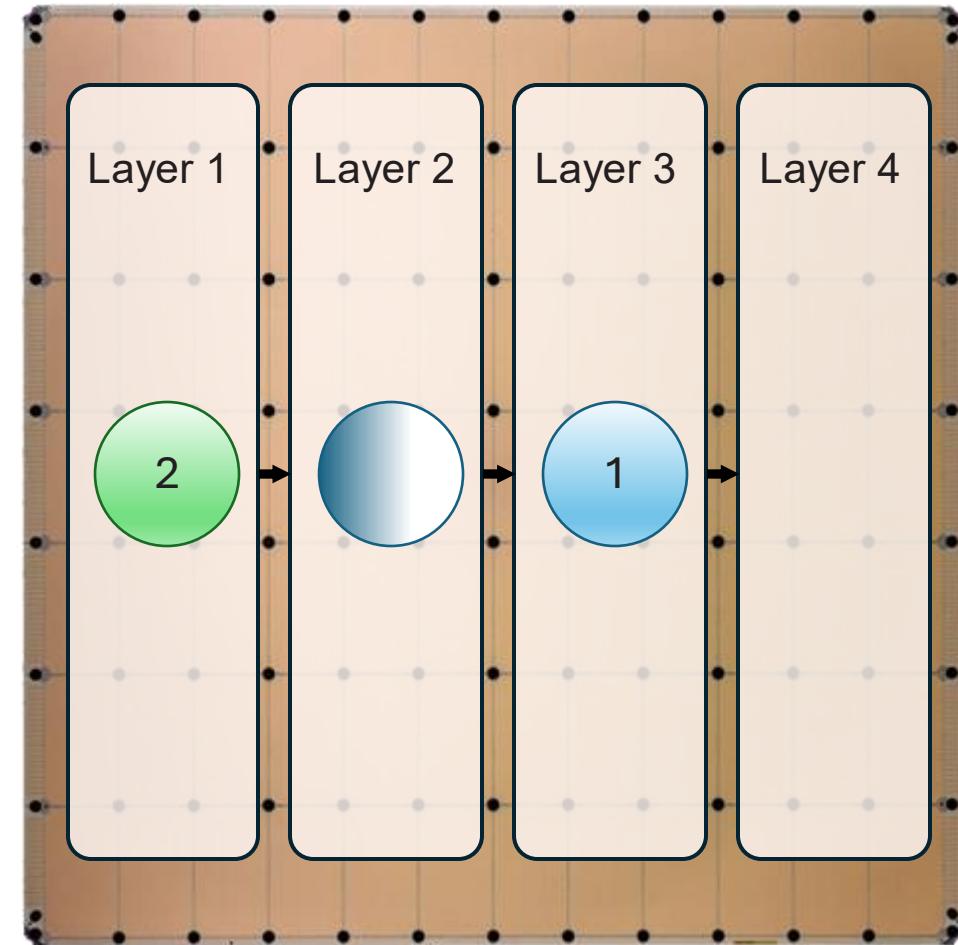
- Input is known upfront so no need to wait for full pipeline execution for next token



# High performance input processing

## Input tokens can be processed in parallel

- Input is known upfront so no need to wait for full pipeline execution for next token
- Run multiple single user input tokens in multiple pipe stages
- Can achieve higher single user input speed using empty pipe stages when fewer users

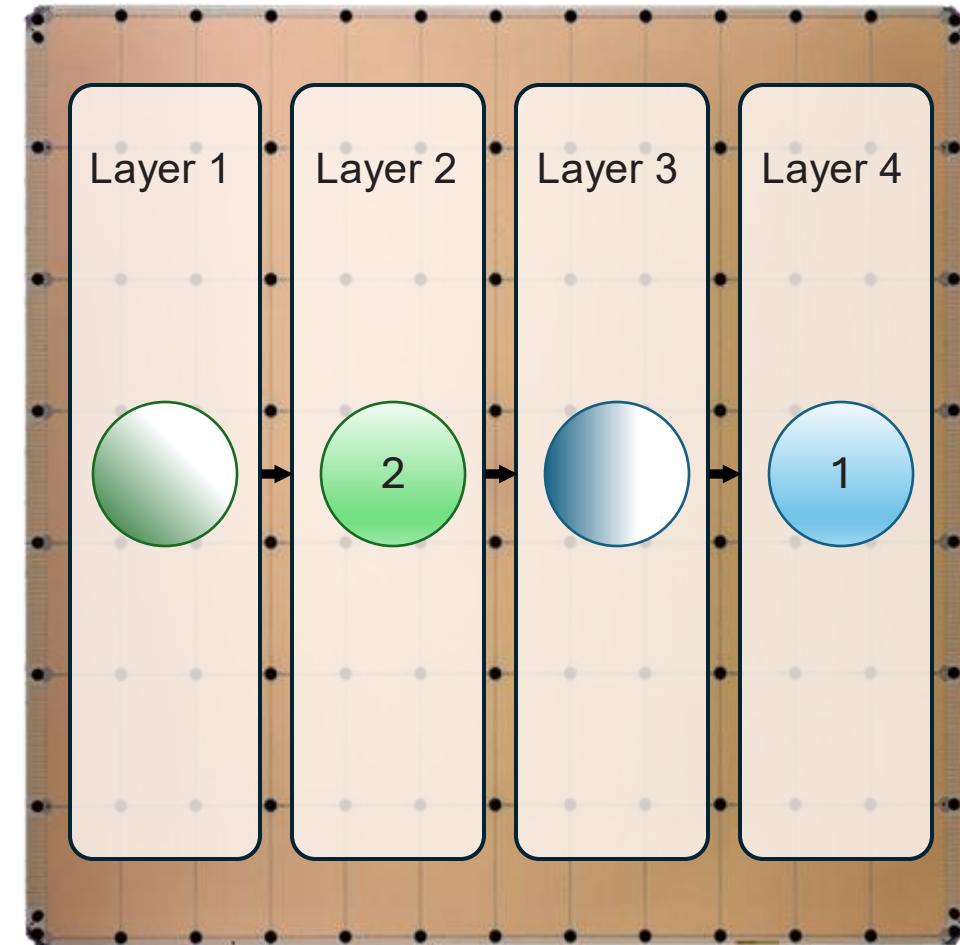


# High performance input processing

## Input tokens can be processed in parallel

- Input is known upfront so no need to wait for full pipeline execution for next token
- Run multiple single user input tokens in multiple pipe stages
- Can achieve higher single user input speed using empty pipe stages when fewer users

Flexible faster input processing driving maximum throughput



# Training Deep Dive

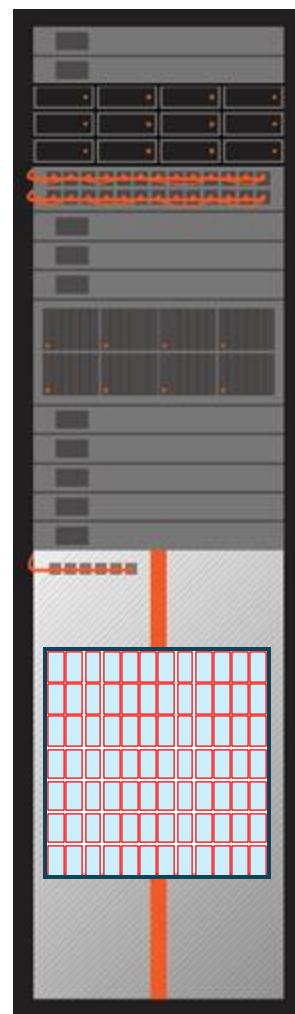
# CS-3 vs. GPU

## Orders of Magnitude Performance Advantage

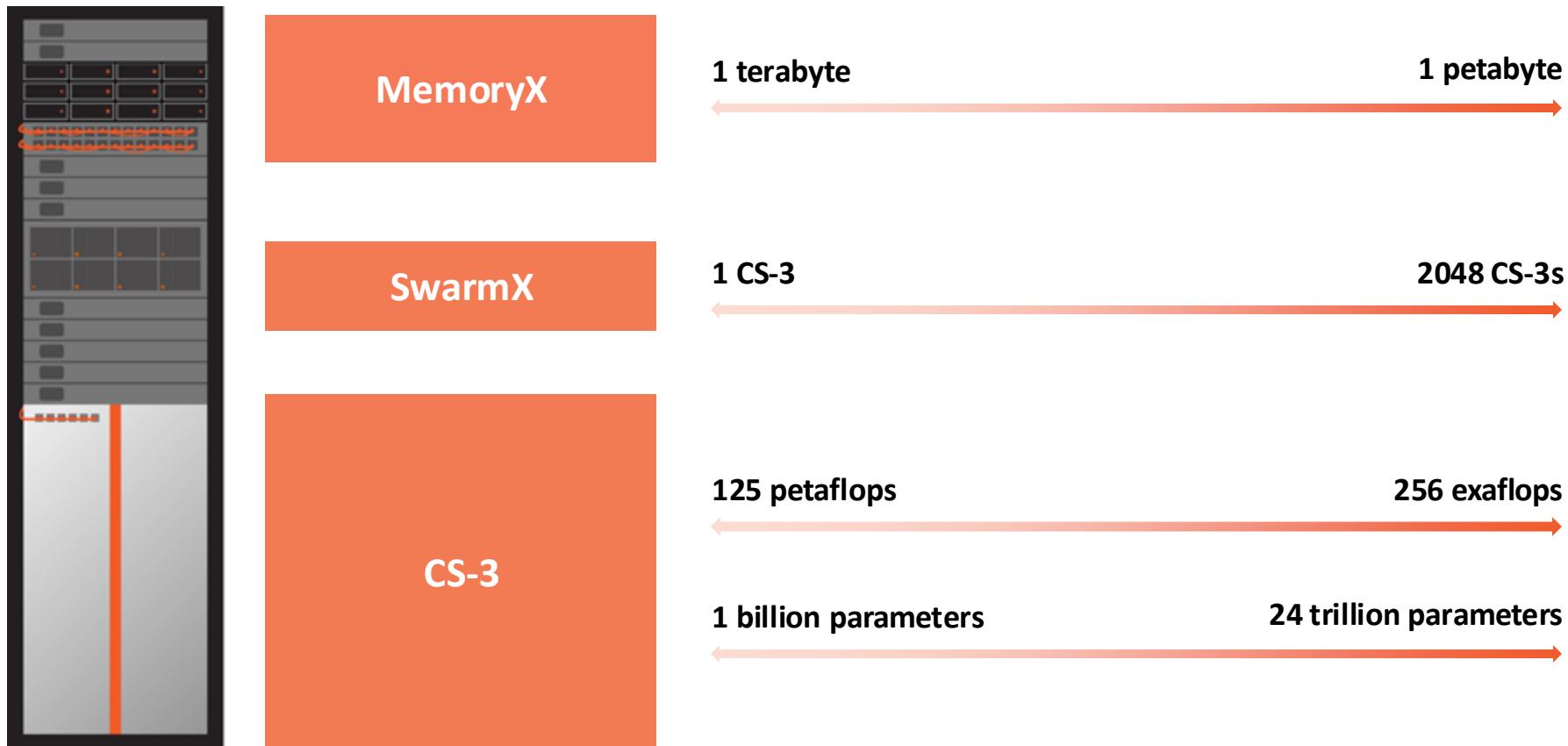
	Cerebras CS-3	Nvidia H100	Cerebras Advantage
Chip size	46,225 mm <sup>2</sup>	814 mm <sup>2</sup>	57x
Cores	900,000	16,896 FP32 + 528 Tensor	52x
On-chip memory	44 Gigabytes	0.05 Gigabytes	880x
Memory bandwidth	25 Petabytes/sec	0.003 Petabytes/sec	7,000X
Fabric bandwidth	245 Petabits/sec	0.0576 Petabits/sec	3,715X

**Enabling large scale training  
Finetune LLaMA 70B on 1B tokens in a day  
on a single chip**

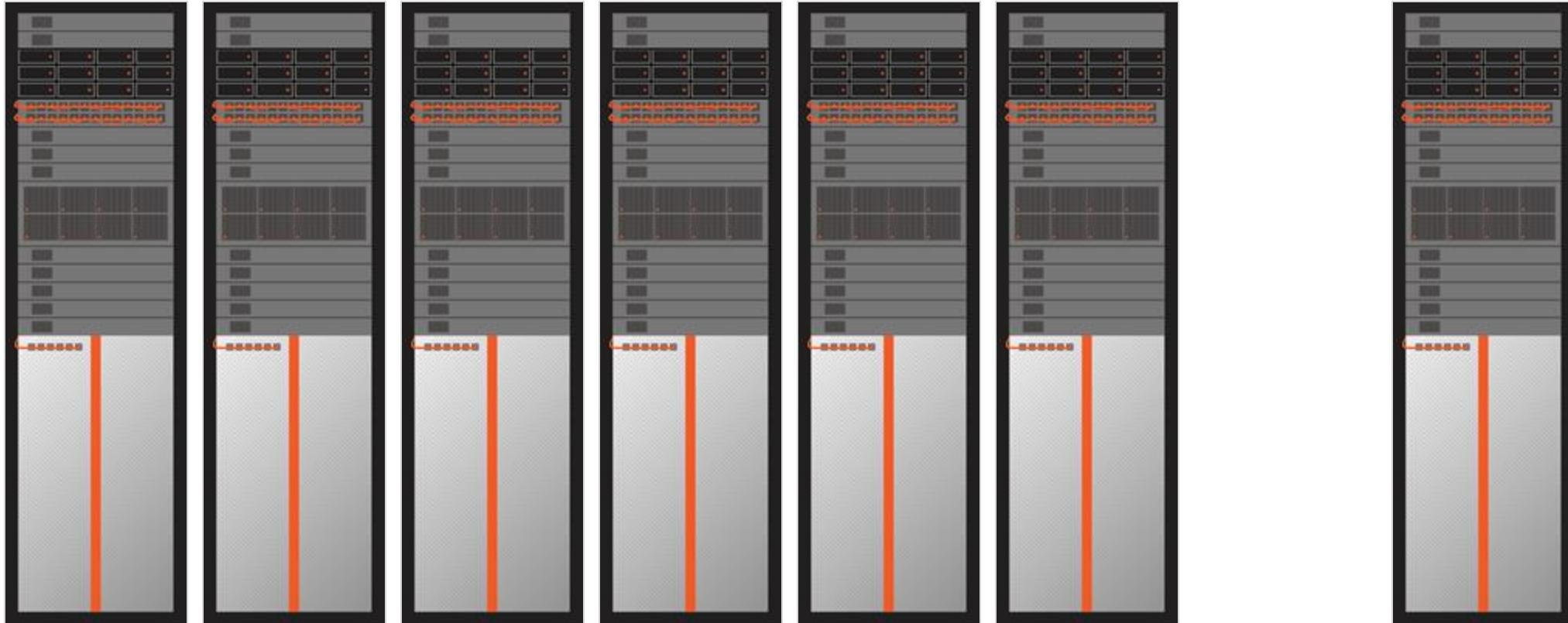
# Wafer Scale Cluster: Purpose Built for Wafer-Scale



# Wafer Scale Cluster: The World's Most Scalable AI Supercomputer



# Exa-scale Performance

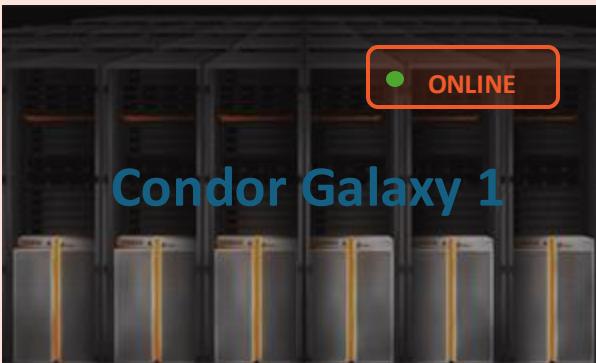




# AI Supercomputers

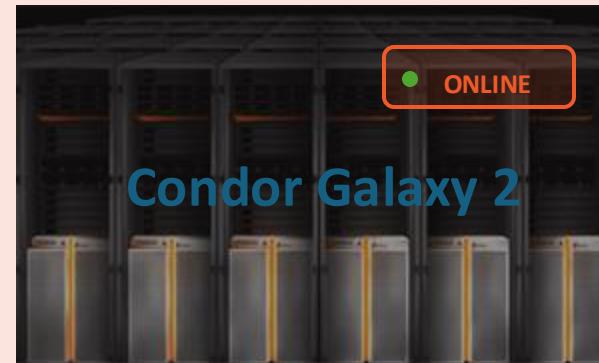
## Built & Operated in the United States

Santa Clara, CA



- 4 ExaFLOPs
- 64 x CS-2s
- 82 TB of Memory

Stockton, CA



- 4 ExaFLOPs
- 64 x CS-2s
- 82 TB of Memory

Dallas, TX



- 24 ExaFLOPs
- 192 x CS-3s
- 324 TB of Memory

# Cluster Components

# MemoryX External Memory

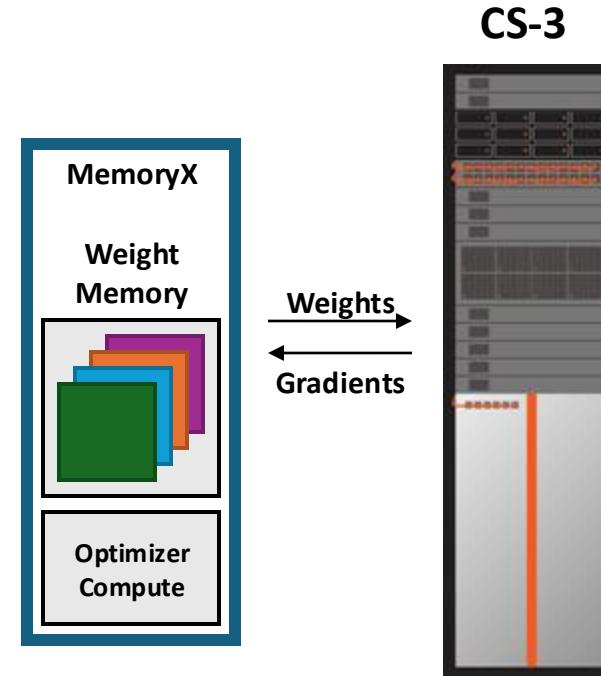
## Virtually Unlimited Model Weight Capacity

### Model capacity not limited by device

- Weights streamed onto wafer to compute layer
- Weights trigger compute using HW dataflow
- Weights are never stored on wafer

### Decoupling weight optimizer compute

- Gradients streamed out of wafer



Memory hierarchy capable of massive models on single device

# Enabling Multi-Trillion Parameter Models

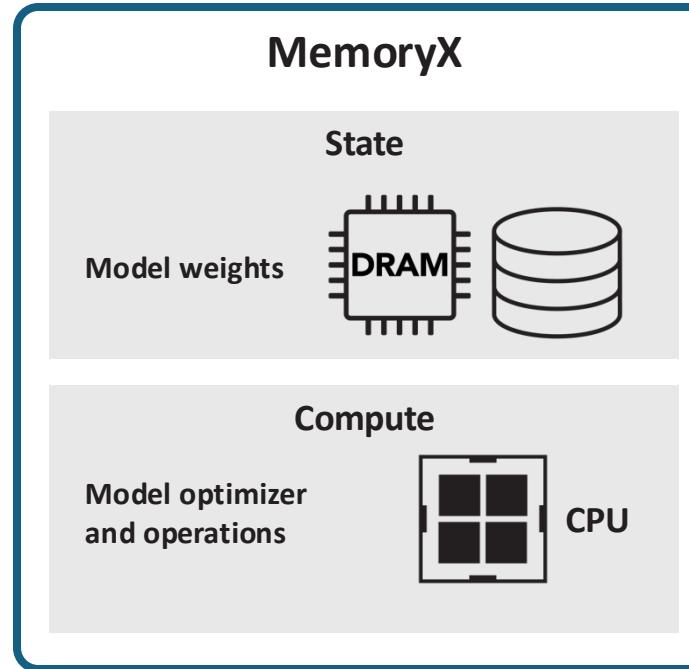
## Most Scalable and Efficient Model Memory

### Efficient hybrid state store

- Weights stored in DDR5 and Flash
- Perf and power/cost efficiency

### Flexible compute

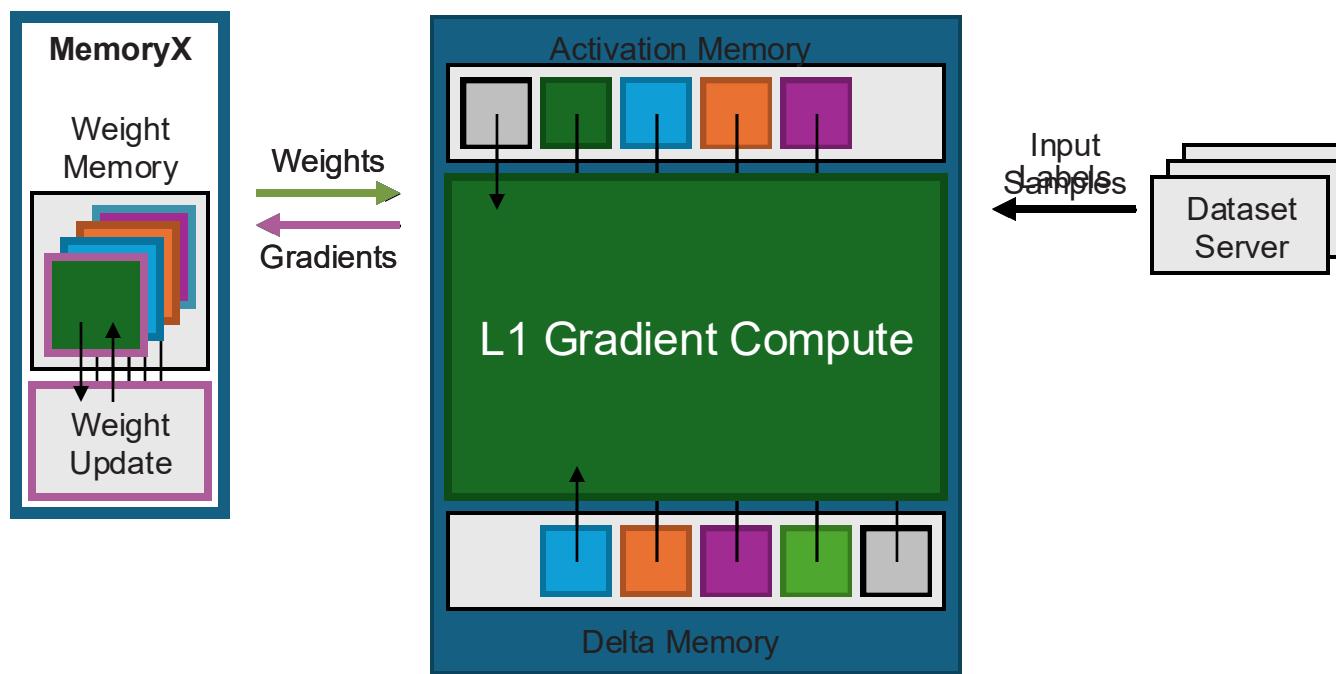
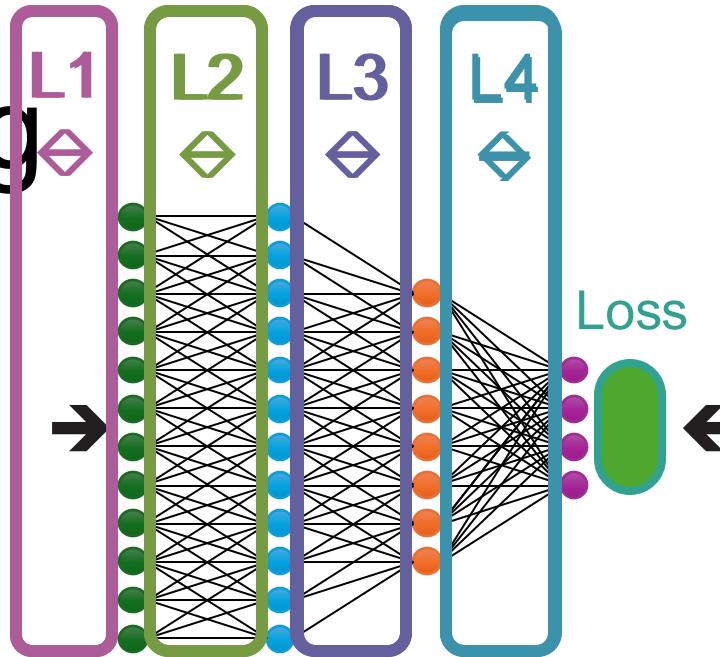
- Optimizer and other ops run on CPU
- General purpose and flexible
- Support for all common ML ops



### Cluster Options

	CS-2	CS-3
DRAM Memory	12 TB DDR4 240B params	<b>36 TB DDR5 720B params</b>
Flash Memory		<b>1.2 PB 24T params</b>
CPU Perf	1x	<b>2x</b>

# Weight Streaming In Action



# SwarmX Fabric

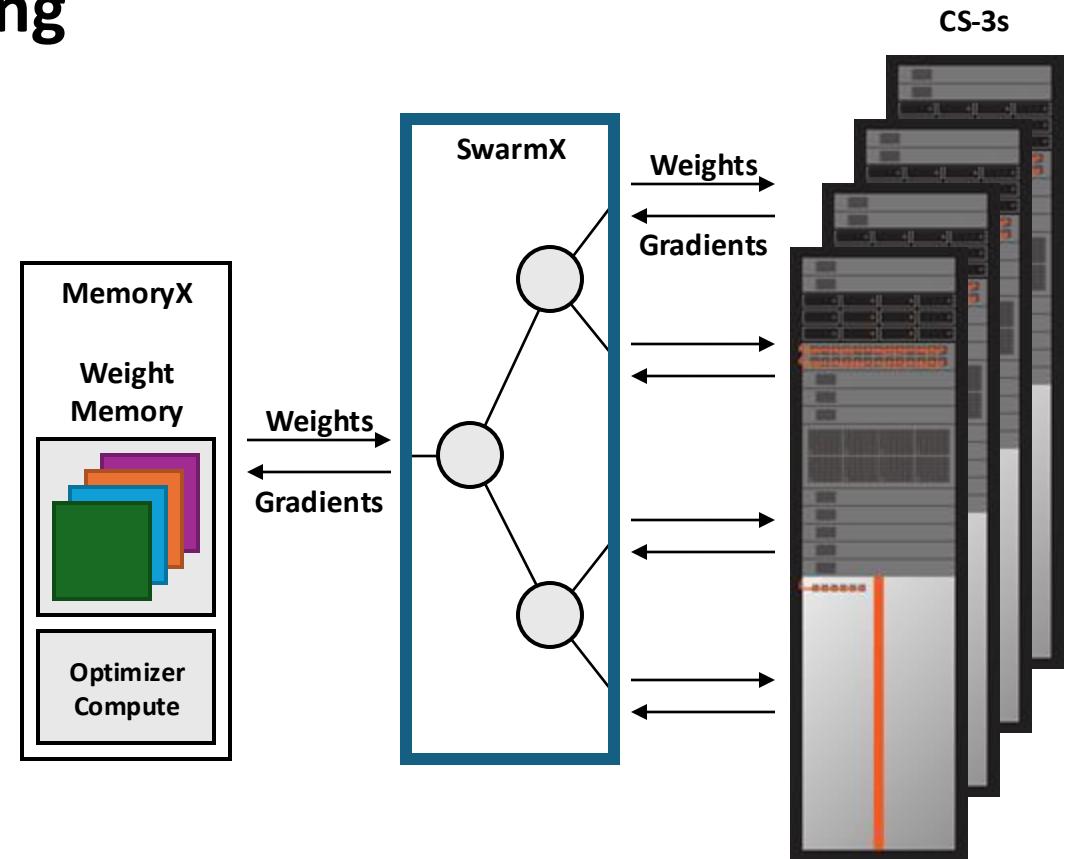
## Purpose Built Interconnect for Simple Scaling

### Data-parallel only training across CS-3s

- Weights are broadcast to all CS-3s
- Gradients are reduced on way back

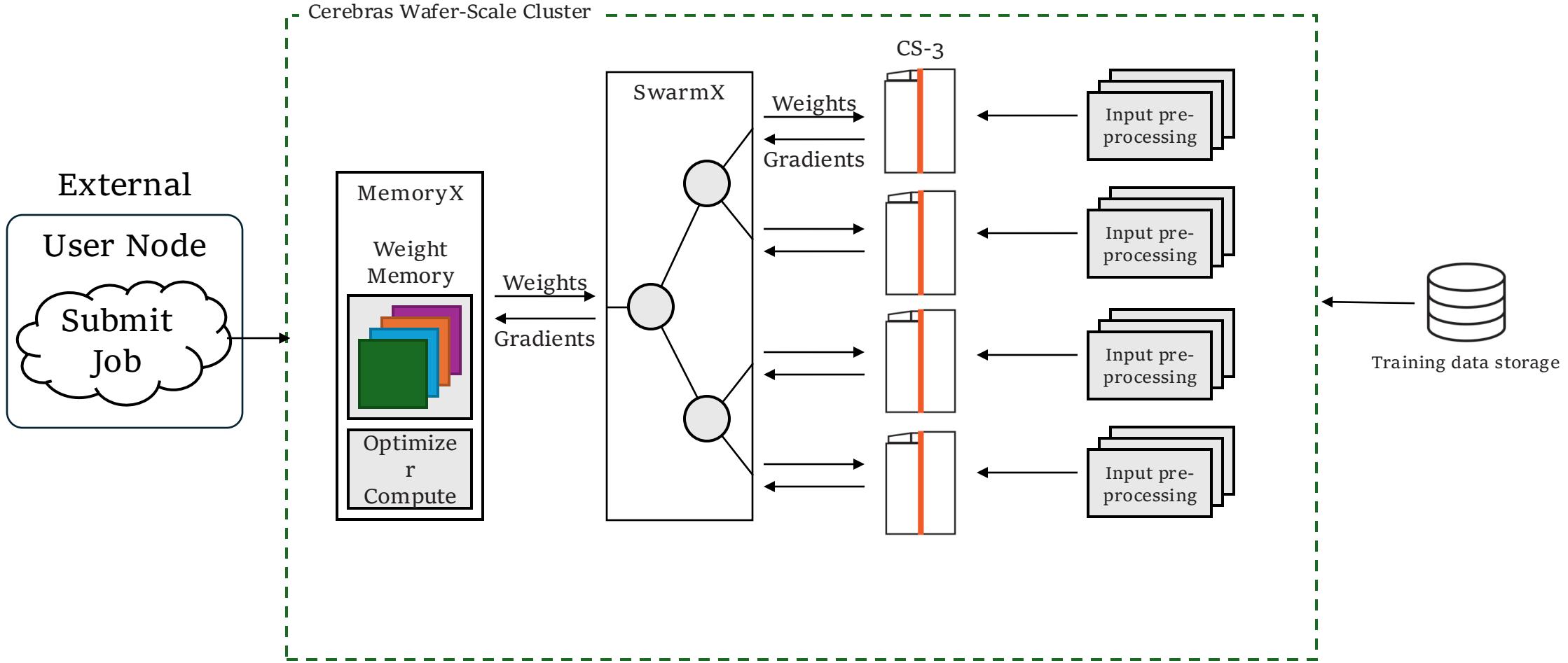
### Multi-system scaling with the same execution model as single system

- Same system architecture
- Same network execution flow
- Same software user interface



Scaling to cluster compute while operating like a single device

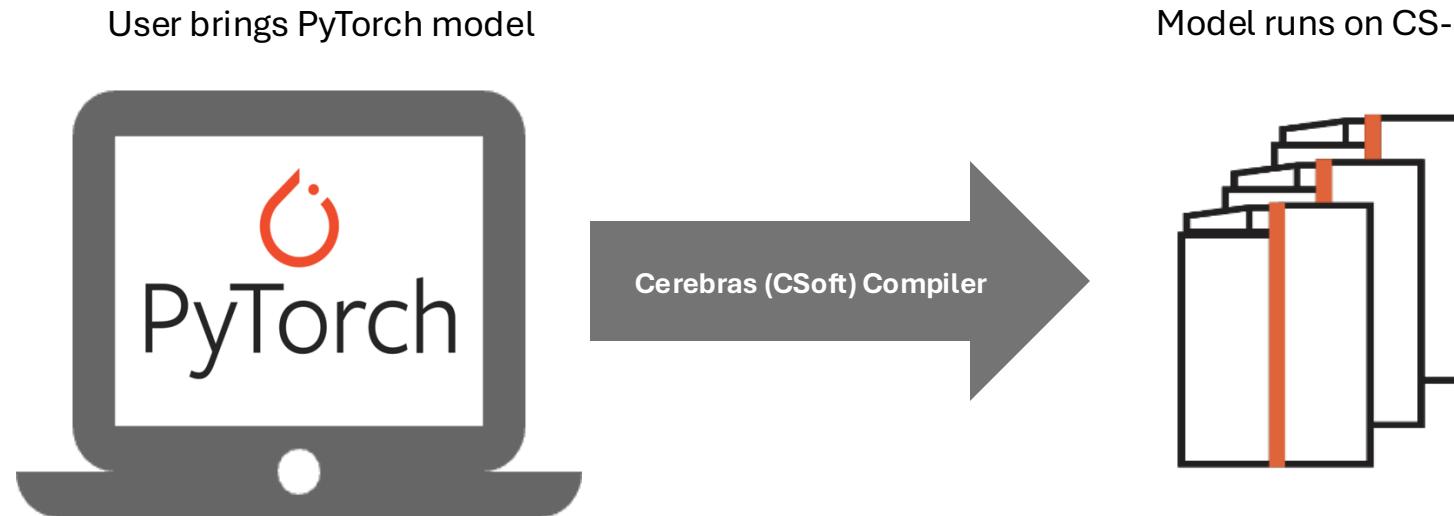
# System Architecture: The Cerebras Wafer-Scale Cluster & Appliance Model



# Programmability

# Models Written for GPUs in PyTorch Run Seamlessly on Cerebras Hardware

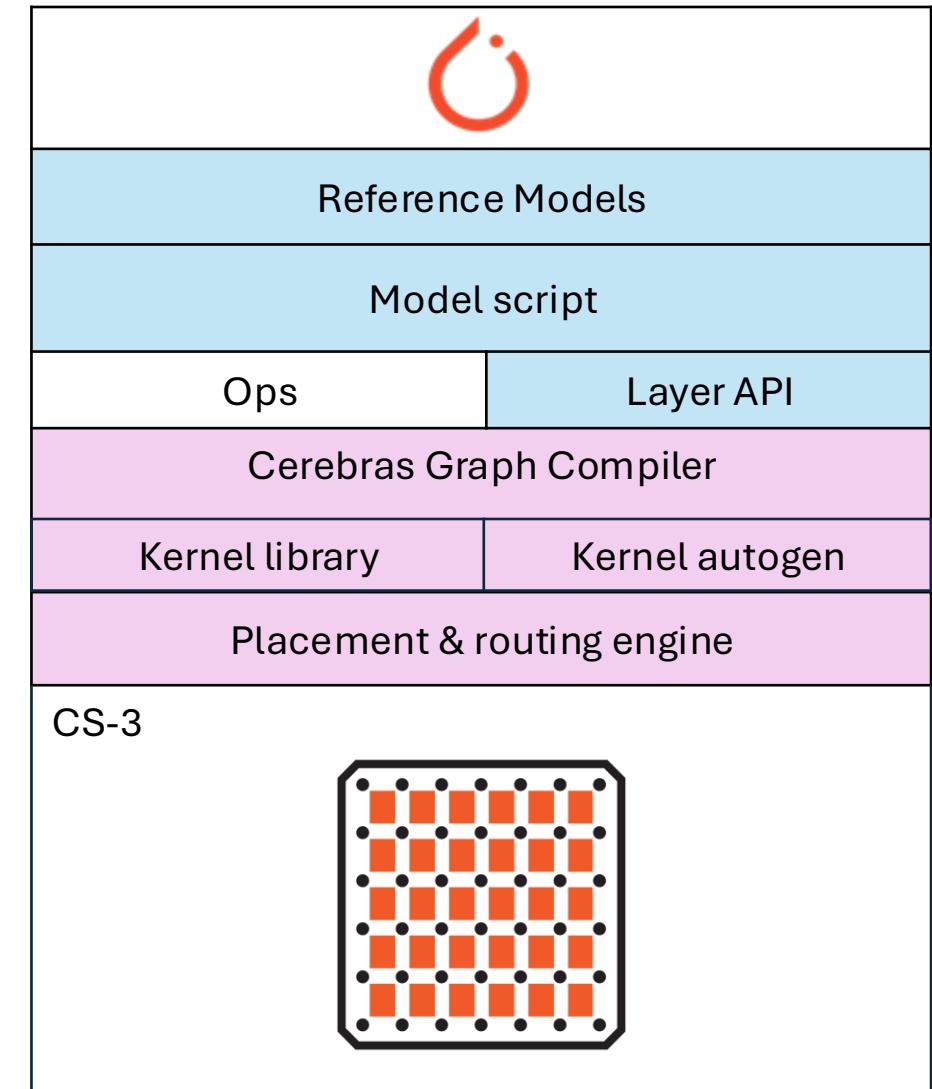
---



# Cerebras Software Workflow: From Model to Wafer

## Integration with PyTorch

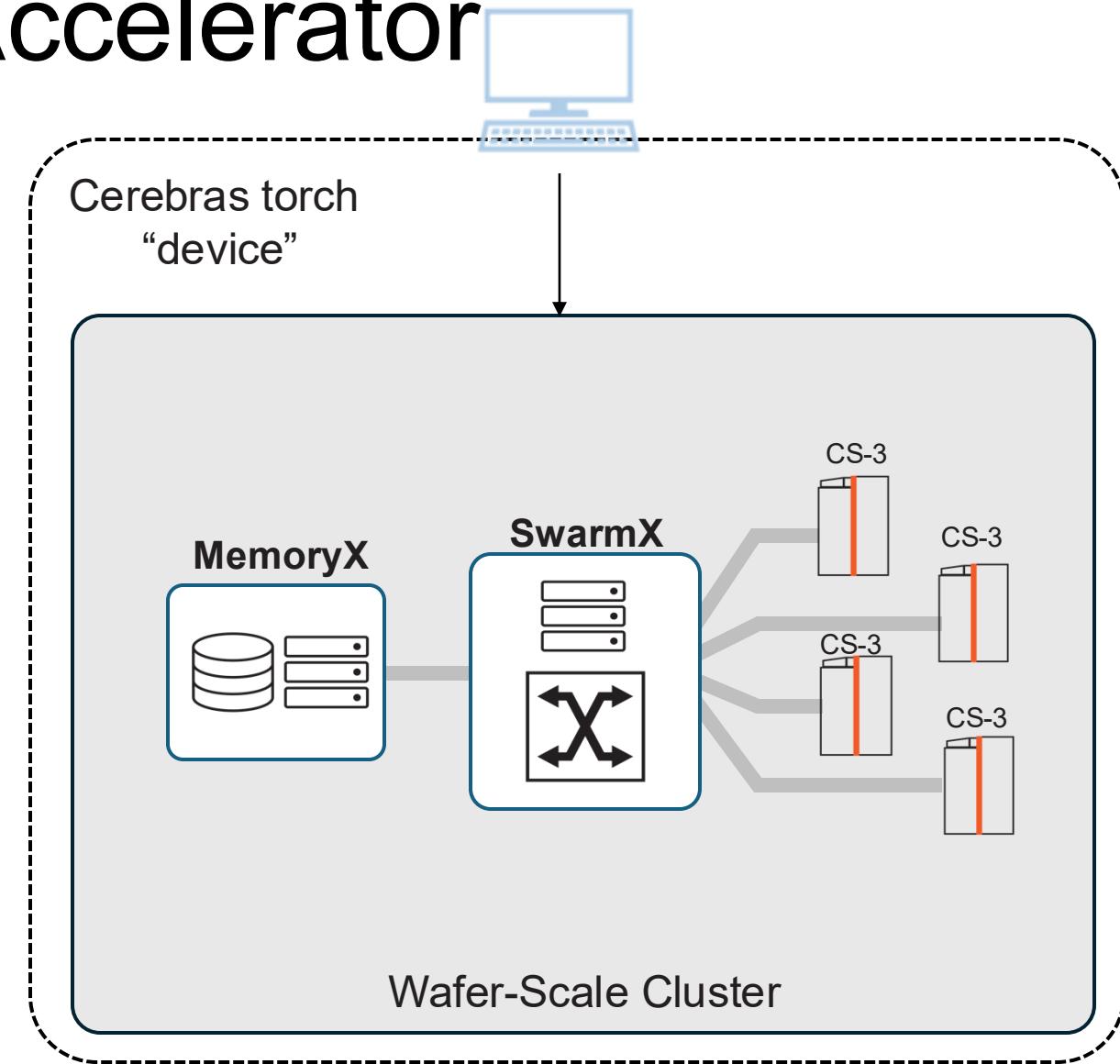
- Models defined in framework + Cerebras API
- Optimally maps from PyTorch to high performance kernels
  - Uses polyhedral code-generation or hand-written kernels
- Compiler using industry standard MLIR framework
  - Cerebras is an active contributor to the MLIR open-source community
- User does not worry about distributed compute or parallelism



# The Cluster *is* the ML Accelerator

## Disaggregation of memory and compute

- Architect cluster-level memory and compute
- Store model weights externally
- Untangle memory and compute dependency
- Cluster acts a remote accelerator
- All complexity of distributed training, data parallelism, scaling are handled automatically by the Wafer-Scale Cluster
- Exposed as a single device to PyTorch



# How to scale from 1B to 70B on Cerebras

gpt3\_1b\_params.yaml

```
### GPT-3 XL 1.3B  
  
hidden_size: 2048  
num_hidden_layers: 24  
num_heads: 16
```

llama2\_70b\_params.yaml

```
### Llama-2 70B  
  
hidden_size: 8192  
num_hidden_layers: 80  
num_heads: 64
```



Training:

```
python run.py \  
--params gpt3_1b_params.yaml \  
--num_steps=100 \  
--model_dir=model_dir \  
 
```

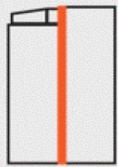


Training:

```
python run.py \  
--params llama2_70b_params.yaml \  
--num_steps=100 \  
--model_dir=model_dir \  
 
```

# Scaling from one CS-3 to a cluster is a 1-line change

```
python run.py  
  --params params.yaml  
  --num_csx = 1 ← How many nodes?  
  --model_dir = model_dir  
  --num_steps = 1000  
  --mode=train
```



# Resulting in Near Linear Scaling

Any Scale While Operating as a Single Device

## Real world seamless cluster scaling

- User: G42
- Model: Jais30B
- Cluster: Condor Galaxy-1
- Experience: “It just worked”
  - No complex distributed software
  - No changes to parallelism model
  - No changes to hyper-parameters

## Training SOTA large models everyday

- Unique capability enabled by wafer-scale

