

A large, light gray decorative graphic on the left side of the slide, consisting of several concentric, curved lines that form a partial circular shape.

# Cerebras CS-2: the AI Compute Engine for Neocortex

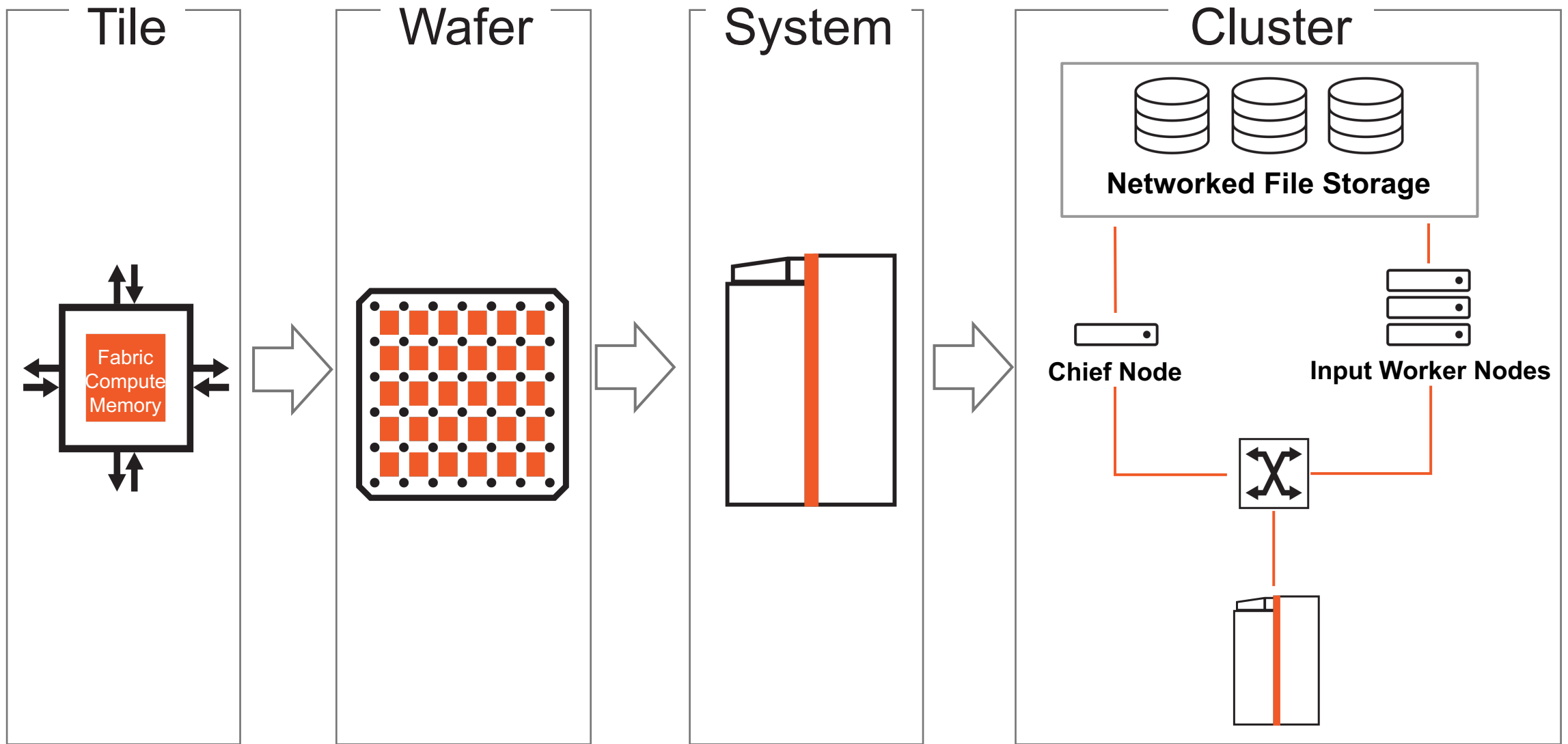
June 1, 2022

# Outline

- CS-2 overview
- CS-2 for Deep Learning
  - PyTorch and TF integration
  - Reference implementations, docs, supported layers
  - Examples of successful DL projects
  - DL topics of interest
- CS-2 for HPC
  - Programming with the Software Development Kit (SDK) and Cerebras Software Language (CSL)
  - Examples of successful HPC projects
  - HPC topics of interest



# CS-2 Overview





# Cerebras Wafer Scale Engine (WSE-2)

## The Most Powerful Processor for HPC & AI

<b>850,000</b>	cores optimized for sparse linear algebra
<b>46,225 mm<sup>2</sup></b>	silicon
<b>2.6 trillion</b>	transistors
<b>40 Gigabytes</b>	of on-chip memory
<b>20 PByte/s</b>	memory bandwidth
<b>220 Pbit/s</b>	fabric bandwidth
<b>7nm</b>	process technology

**Cluster-scale performance in a single chip**

# Cerebras CS-2 System

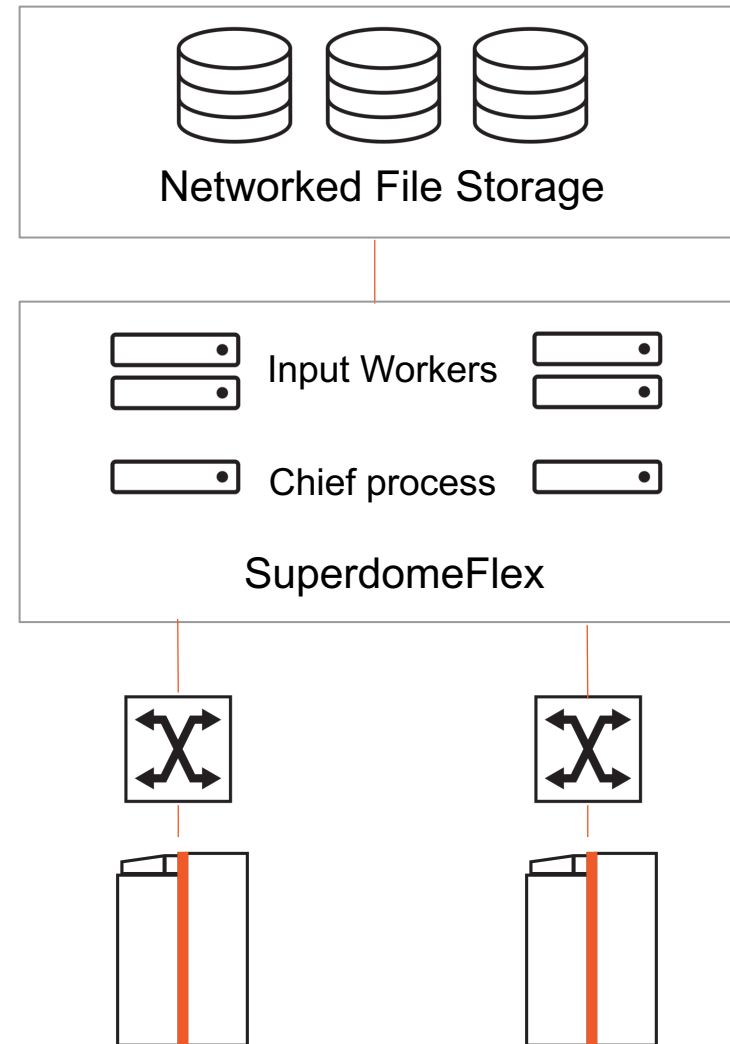
## The world's fastest AI & HPC accelerator

- ✓ Deploy easily into existing racks
- ✓ Cluster-scale in a single system
- ✓ Datacenter-scale in a cluster
- 15 RU, standard data center rack
- 12 x 100 GbE IO
- Closed-Loop Water Cooling
- 28 kW max power



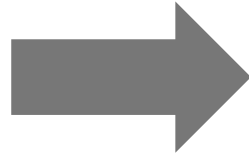
# Deployment and job execution within Neocortex

- CS-2 is a **network-attached accelerator**
- Cerebras SW runs on CS-2 and on the SuperdomeFlex
  - Chief compiles the code and manages CS-2
  - Input workers read data, run the input pipeline, and stream data to CS-2
- Loss output, summaries, checkpoints are streamed from CS-2 to SuperdomeFlex
- Jobs are managed by slurm

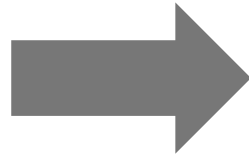
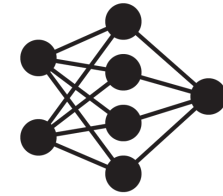


# Developer Resources

For ML  
developers



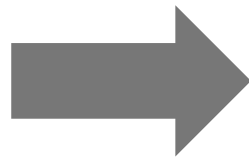
Cerebras Implementations  
(Public References and Private Model Zoo)



Cerebras ML Software



For kernel  
developers



Cerebras SDK







# CS-2 for Deep Learning

# Frameworks supported



**TensorFlow**

**Class:**

**CerebrasEstimator**

- Based on TF Estimator, takes over executions after XLA compilation
- TensorFlow 2.2



**PyTorch**

**Python Module:**

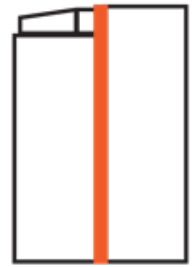
**cerebras.framework.torch**

- Based on PyTorch XLA
- Wrappers for Dataloader, Module, Session
- PyTorch 1.11

# How do we translate a model into a CS executable?

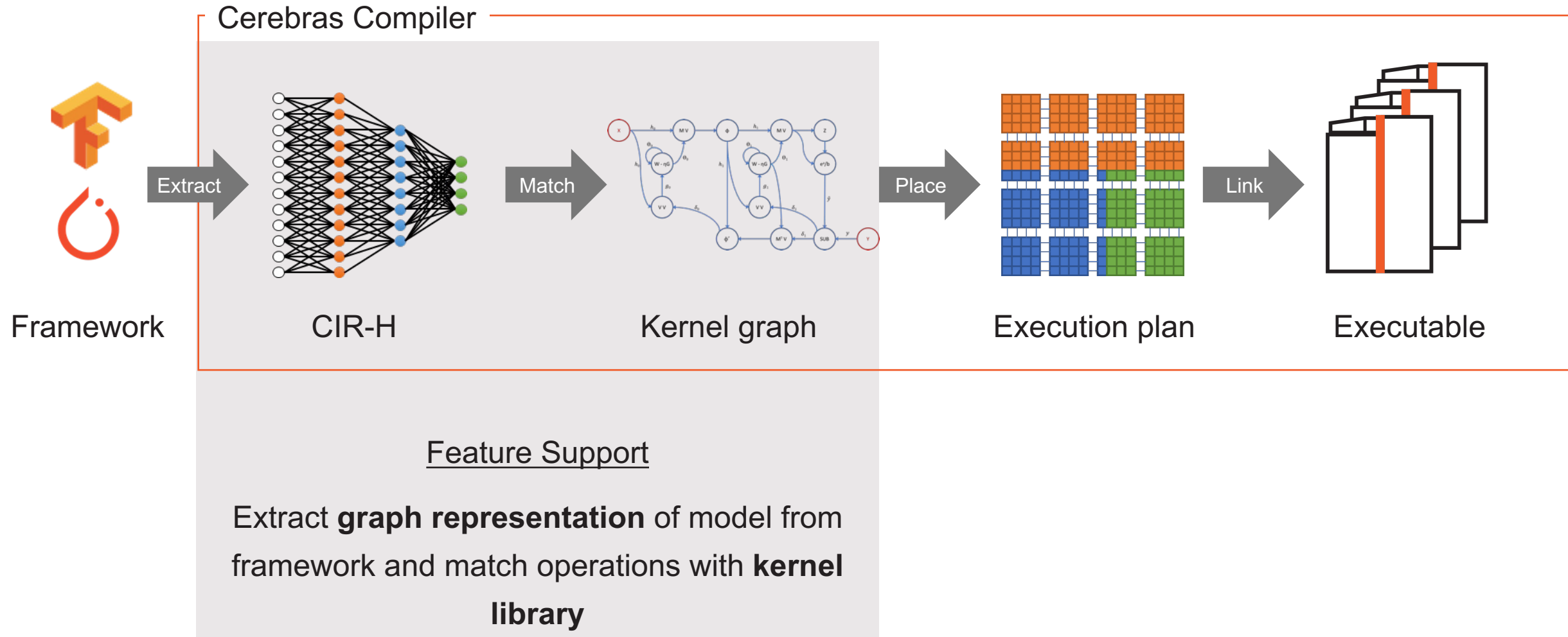


Framework

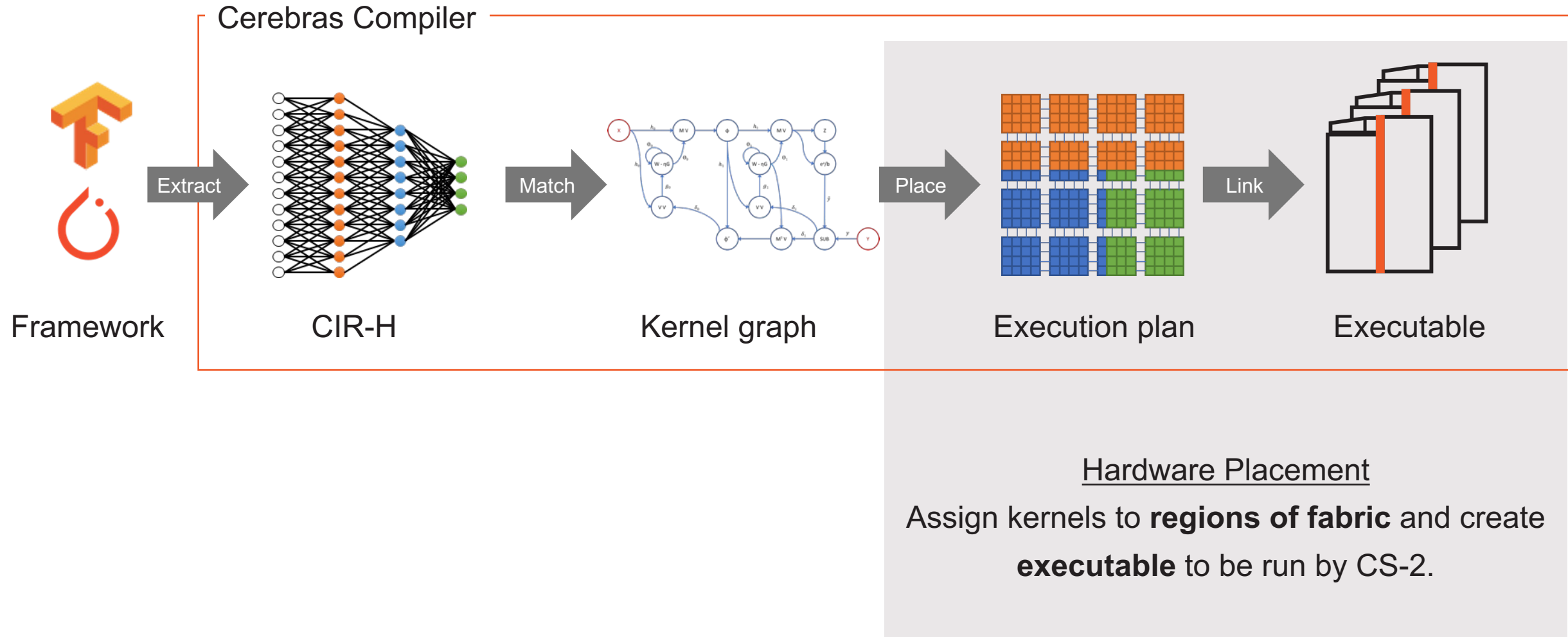


CS-2

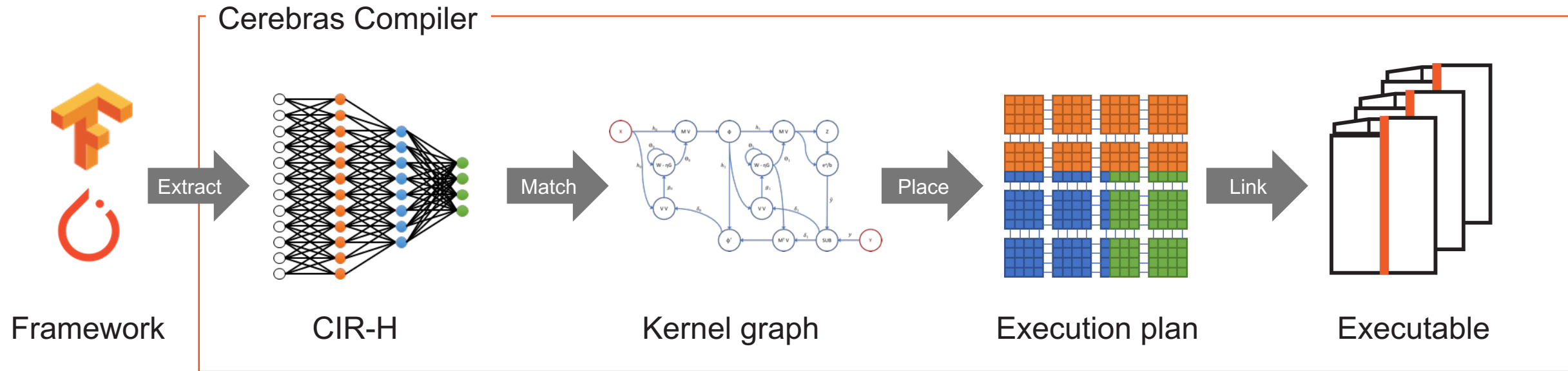
# The Cerebras Software Platform



# The Cerebras Software Platform



# The Cerebras Software Platform



Program a cluster-scale resource with the ease of a single node

# ML Software Key Features

## Network Architectures

- Transformers (TF and PyT)
  - E.g., BERT, RoBERTa, AIAYN, T5, GPT
- Multi-layer Perceptrons (MLP) (TF and PyT)
- Experimental (in Beta, TF only)
  - Recurrent Neural Networks (LSTMs)
  - Graph Neural Networks (GNN's)
  - Convolutional Variational Autoencoders
  - UNet

## General features

- Supports Train, Eval and Predict
- Trained weights in standard TF and PyT formats
- Monitor your runs with TensorBoard
- Automatic input pipeline optimization
- Automatic kernel generation for non-linears and losses
  - E.g., SeLU, Softmax Cross Entropy, etc.
- Multi-replica support for smaller models

Poll: What types of models are you working with?



# Resources

Documentation : docs.cerebras.net



Search the docs ...

Software Release Notes

Documentation Updates

## CEREBRAS BASICS

How Cerebras Works

The Cerebras ML Workflow

## GETTING STARTED

Checklist Before You Start

TensorFlow Quickstart

PyTorch Quickstart

## MODEL ZOO

Model Support Matrix

## DEVELOP WITH TENSORFLOW

Develop With TensorFlow

DEVELOP WITH PYTORCH

## Explore the Documentation

This documentation will help you program for the CS system. It covers both basic and advanced topics. Use these docs to accelerate your machine learning training and inference applications on the CS system. Here you will find getting started guides, quickstarts, tutorials, code examples, release notes, and more.

### Learn Cerebras basics

#### Big picture view of a CS system

[How Cerebras works](#)

Start with this big picture before you dive into your ML development with Cerebras system.

[Programming model and the compiler](#)

Get to know how Cerebras separates compile vs execution, and the compiler flow from framework to the executable.

[The Cerebras CPU cluster](#)

How a Cerebras multi-worker configuration differs from a GPU multi-worker configuration.

#### ML workflow on Cerebras

[Cerebras ML workflow](#)

Whether your framework is TensorFlow or PyTorch, get to know the general ML workflow on

Cerebras Reference Implementations

The screenshot shows the GitHub repository page for Cerebras/cerebras\_reference\_implementations. The repository is public and has 2 watchers, 0 forks, and 0 stars. The main branch is 'main' with 1 branch and 0 tags. The repository contains a README.md file and several subdirectories: bert, common, and fc\_mnist, all added 1.2.0 code 2 days ago. The README.md file is displayed, showing the title 'Cerebras Reference Implementations'. The right sidebar shows the repository's metadata, including the README, 0 stars, 2 watchers, and 0 forks. The releases section shows no releases published. The packages section shows no packages published. The contributors section shows 2 contributors: noah-cerebras (Noah Arthurs) and aviv-cerebras (Aviv Yaffe). The languages section shows a bar chart with Python at 99.8% and Shell at 0.2%.



# BERT on PubMed



## Objective:

Retrain BERT Base and train BERT Large from scratch on PubMed abstracts + PMC full texts  
13.9B words: **4.2x larger than BERT dataset**



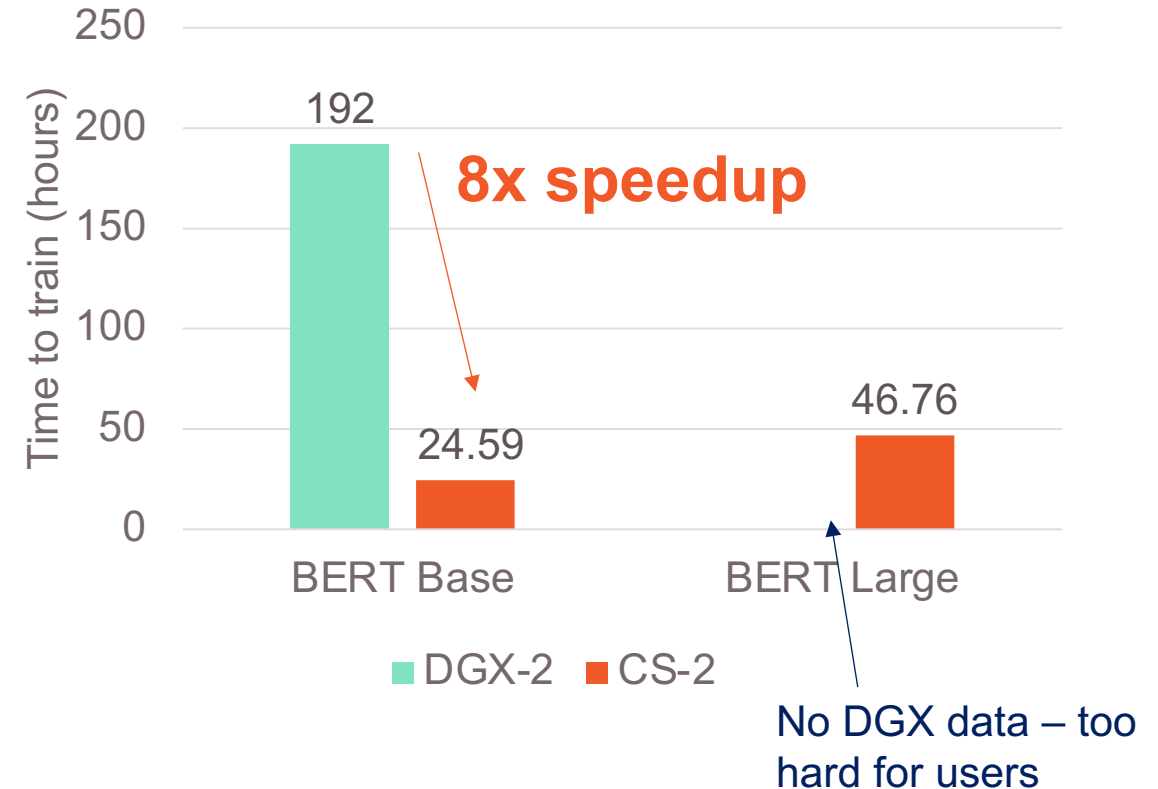
## Target:

Converge to published accuracy  
(Eval metrics after pre-training and accuracy on BLURB NER and BioASQ after fine-tuning)



## Challenge:

Very long time-to-train  
Difficult for team to iterate  
High complexity involved with building larger GPU cluster



**DGX scaling is sublinear. Requires > 10 systems to achieve 8x acceleration.  
~4x faster to train Bert LARGE on CS-2 than Bert BASE on DGX.**

# Epigenomic Language Model Developed by GSK



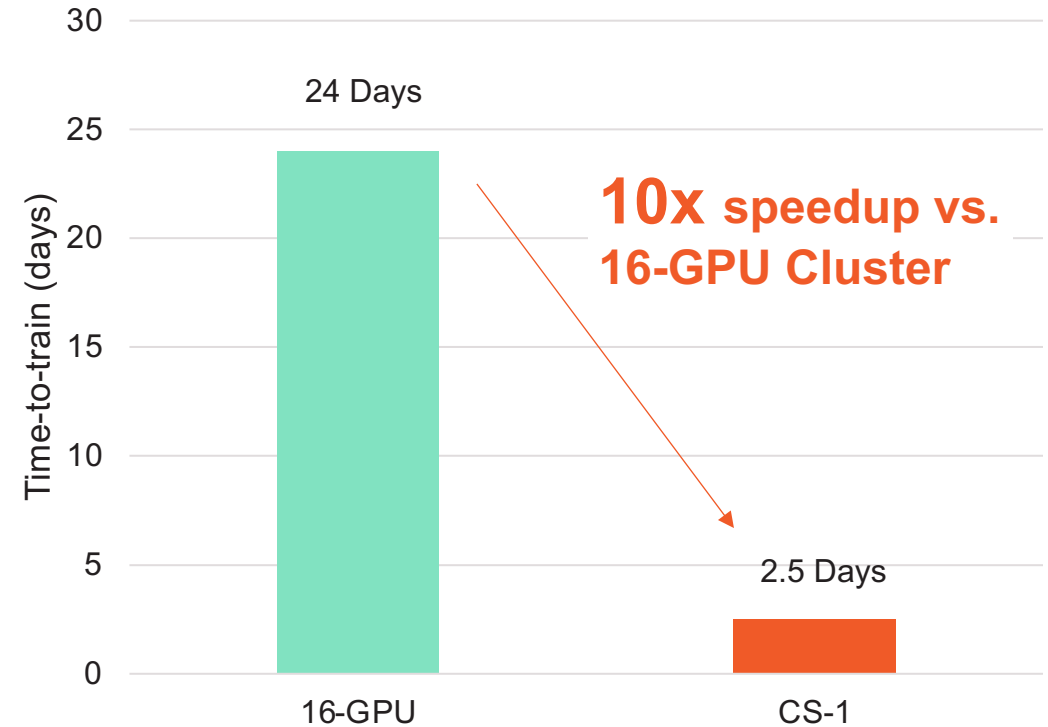
**Objective:** Accelerate genetic validation of drug targets using novel technique that includes epigenomic data in NLP models, rather than genome-only models



**Challenge:** Training this complex model with massive datasets would take several weeks on a 16-GPU cluster, making rapid experimentation impractical



**Outcome:** ~10X training speedup empowered researchers to experiment with epigenomic data and demonstrate superior results to DNA-only datasets



“The training speedup afforded by the Cerebras system enabled us to explore architecture variations in a way that would have been prohibitively time and resource intensive on a typical GPU cluster”

— “Epigenomic Language Models Powered by Cerebras”, Dec 2021. [arxiv.org/abs/2112.07571](https://arxiv.org/abs/2112.07571)



# BERT on Pfam for TAPE



## Task:

Explore semi-supervised NLP models to learn representations for proteins, evaluate with Tasks Assessing Protein Embeddings (TAPE)



## Target:

Converge to published accuracy on downstream tasks



## Challenge:

Very long time-to-train  
Difficult for team to iterate  
High complexity involved with building larger GPU cluster



## Outcome:

CS-1 reduced training time to from a week to less than a day  
The system enables dramatic acceleration in experimentation pace

Poll: What are the typical model sizes you work with?

# Topics of interest for ML applications

- Domain-specific large-scale transformer-style language models
- Transfer learning with large-scale transformer-style language models
- Multilingual models, machine translation
- Learning representations for DNA and RNA sequences, proteins
- Training extreme-scale language models with sparse weights
- Training transformer-style models with long sequences
- Scaling laws, efficient training regimes for transformer models
- Image segmentation with high-resolution images



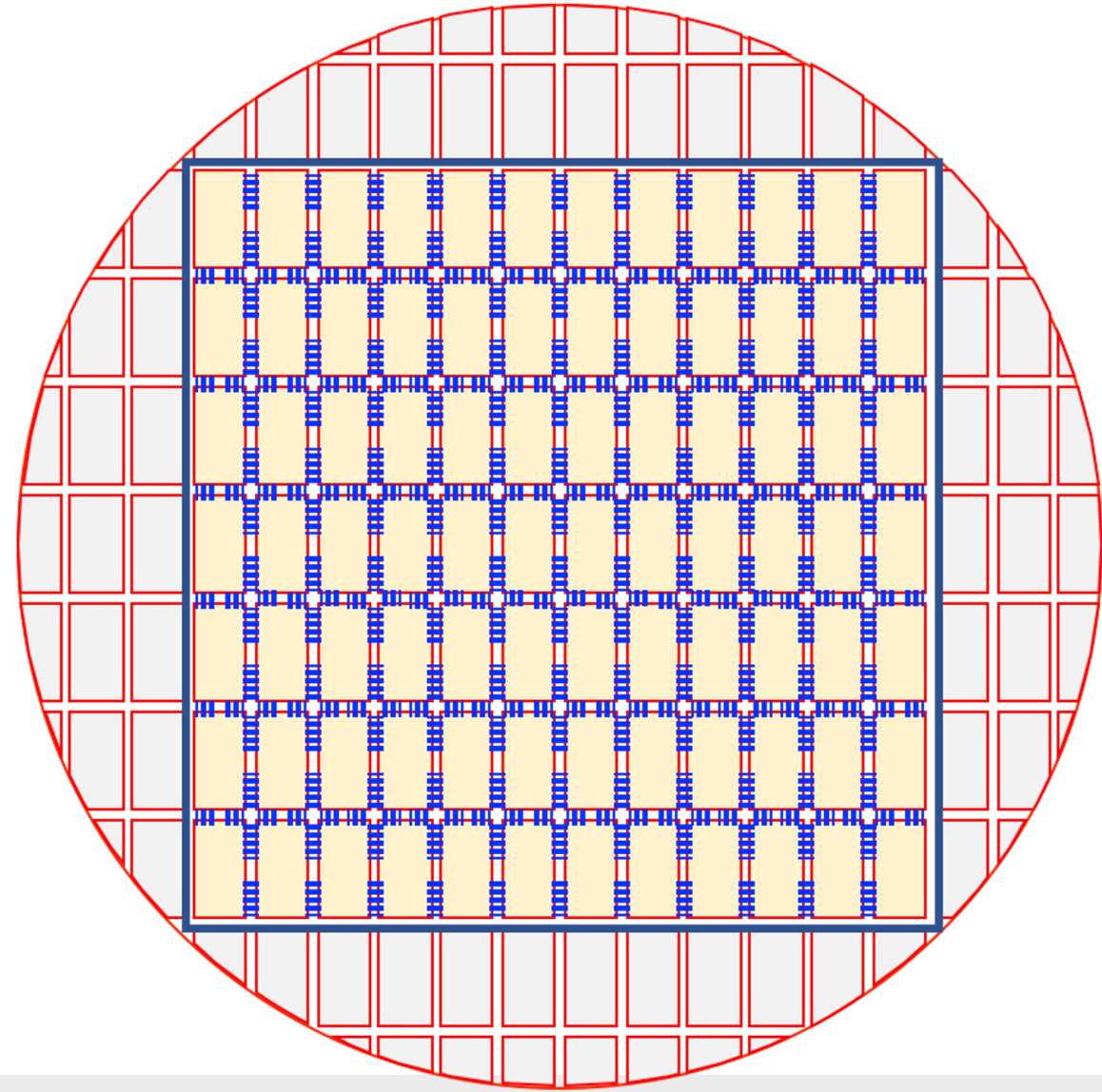
# CS-2 for HPC

# Does your application **scale poorly across nodes**?

**Examples:** *FFT-based solvers, particle simulators, non-linear problems with iterative solvers*

## **The Cerebras solution:**

- The WSE-2 has a fabric that is **high bandwidth** and **low-latency**, allowing for excellent parallel efficiency for non-linear and highly communicative codes
- The CS-2 system has **850k cores** and can fit problems on an individual chip that take tens to hundreds of traditional small compute nodes.
  - Each core is individually programmable





# Is your application **constrained by data access?**



**Examples:** *Stencil based PDE solvers, linear algebra solvers, signal processing, sparse tensor math, big data analysis*

## **The Cerebras solution:**

- The CS-2 system has **40 GB of SRAM** uniformly distributed across the wafer that is **1 cycle** away from the processing element
  - Speeds up memory access by orders of magnitude
- The CS-2 system is capable of **1.2 Tb/s bandwidth** onto the chip
  - Stream data onto the chip as required

Poll: What type of HPC algorithms do you use?

# Cerebras SDK

Language

CSL: Cerebras Software Language

Host APIs with Python

Libraries

Optimized primitives

Tools

Simulator

Debugger

Performance profiler

Visualization

The screenshot displays the Cerebras SDK GUI with several panels:

- Current folder:** <filepath containing artifacts used in the GUI> [SUBMIT]
- Colors:** A list of color-coded inputs: 1 x\_in (orange), 2 Ax\_out (purple), 3 y\_out (red), 4 b\_in (blue).
- Neural Network Visualization:** A grid of nodes connected by lines, with a central node highlighted by a black box.
- Symbols:** A table listing symbols and their types:

Name	Type
A	NOTYPE
Ax_temp	NOTYPE
memcpy	NOTYPE
memset	NOTYPE
memcpy	FUNC
- Instruction Trace:** A table showing execution details:

Cycle	OP Addr	OP Name	Dest	Src0
344	0x3120	s class	0x0 (0x38b7)	0x0 (0x3040)
- Source Code:** A code editor showing Cerebras Software Language (CSL) code:

```
1 var global: i16 = 0;  
2  
3 color main_color = 0;  
4 color output_color = 1;  
5 const dsd = @get_dsd(fabou_t_dsd, {fabric_color =  
  output_color, extent = 1});  
6  
7 task main_task(wavelet_data: i16) void {
```
- Wavelet Trace:** A table showing wavelet activity:

Cycle	Color	Ctrl	Link	Header
3	3	0	W	0x0000
1890	3	0	E	0x0000
1000	2	0	E	0x0000

Copyright © Cerebras 2021

A general-purpose parallel-computing platform and API allowing software developers to write custom programs (“kernels”) for Cerebras systems.

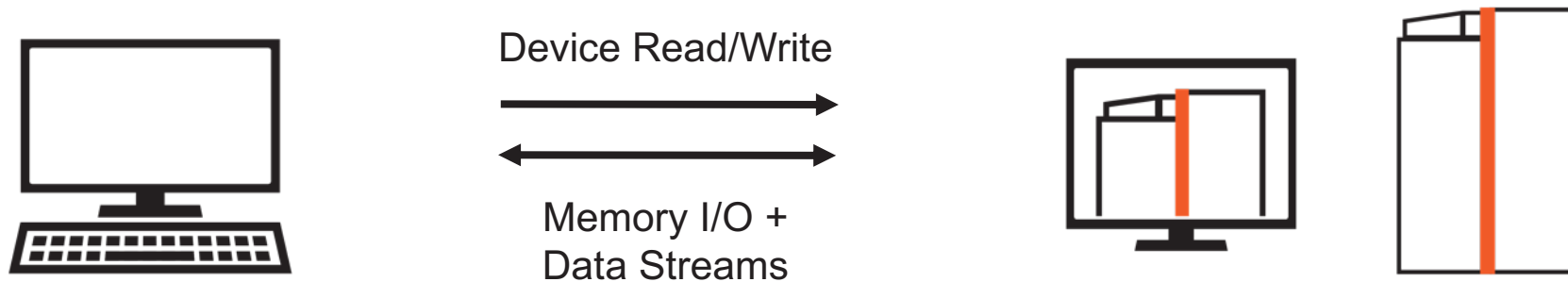
# From a Programmer's Perspective

## Host CPU(s): Python

- Loads program onto simulator or CS-2 system
- Streams in/out data from one or more workers
- Reads/writes device memory

## Device: CSL

- Target software simulator or CS-2
- CSL programs run on groups of cores on the WSE, specified by programmer
- Executes dataflow programs



# Simulation Debug Tools

The screenshot displays the Cerebras SDK GUI interface. At the top, the title bar reads "cerebras SDK GUI". Below it, the "Work directory" is shown as "/cb/cold/swaminathan/testartifacts/sdk/0.4/vijayth2/cslang\_samples\_18\_filters/artifacts".

The main area features a 6x6 grid of Processing Elements (PEs). A black box highlights the PE at coordinates (5, 2). Above the grid, the "Enter PE Coordinate" field is set to "0, 0" and the "Cycle Range" is "0 -".

On the left, a "Colors" panel lists the following: **Timeline** (checked), **x\_in (1)**, **Ax\_out (2)**, **y\_out (3)**, and **b\_in (4)**.

The bottom section contains a "Timeline" view with a horizontal axis from 0 to 962. A red bar indicates a selected cycle range from 163 to 561. Below the axis, a Gantt chart shows activity for various PEs: 4,1,C3; 4,2,C3; 5,2,C3; **4,3,C3** (highlighted); 5,3,C3; and 4,4,C3. A legend at the bottom identifies colors: Active (green), Delayed (yellow), Backpressure (orange), and Control wavelet (blue).

On the right, a "Debug" panel provides details for the selected PE: **PE: x=4 y=3 Color: 3**, **Cycle: 163-561**, **Wavelet ID: 26938034880672**, **Status: Backpressure**, **Direction: N**, **Control: No**, **Index: 0x0000**, and **Data: 0x0000**.

At the bottom right, the status bar shows "CS1 [6 x 6] ALL" and "SELECTED PE: [ 5 , 2 ]".

Copyright © Cerebras 2022

CS1 [6 x 6] ALL SELECTED PE: [ 5 , 2 ]

# Private Documentation: sdk.cerebras.net

**cerebras**

## SDK Documentation

Search the docs ...

- SDK Release Notes
- Documentation Updates
- START HERE**
  - A Conceptual View
  - Kernel Development Flow
- QUICKSTART**
  - Installation and Setup
  - Quickstart
- DEVELOPMENT GUIDES**
  - Working With Code Samples
  - CSL Code Examples
  - CSL Language Guide
- DEBUGGING**
  - Debugging Guide
  - Route Visualizer
- API REFERENCE**
  - SDK API Reference

## Documentation for Developing with CSL

This is the documentation for developing kernels for Cerebras system. Here you will find getting started guides, quickstarts, tutorials, code examples, release notes, and more.

- Start Here**

Computing with Cerebras

[A conceptual, "mental model" view.](#)
- Quickstart**

Compile and run

[Quickstart with a single PE or multiple PEs.](#)
- Kernel Development Flow**

Steps to develop your kernel

[Define layout, assign code to PE and configure routes and colors.](#)
- Working with Code Samples**

Learn how to run the code samples

[A glimpse into the run script.](#)
- Program the WSE**

CSL examples

[Manipulate sparse tensors, configure fabric switches and more.](#)
- Debug**

Learn how to use the debugger

[Trace the instructions, monitor the tasks at a specific PE and trace wavelets.](#)
- Visualize the Fabric**
- SDK API Reference**
- Using CSL**

# Examples Included in the SDK

- Basic tasks and colors
- Multiple source files
- Multi-PE kernel
- Basic parameters
- Wavelet-triggered Tasks
- Arrays and Pointers
- Sparse Tensors
- Memory DSDs
- Fabric DSDs
- Reduction
- Basic Branches
- Initializers
- Modules
- Loops
- Kernel Parameterization
- Fabric Switches
- GEMV
- FFT
- Stencil (Finite Differences)
- Shift-Add Multiply

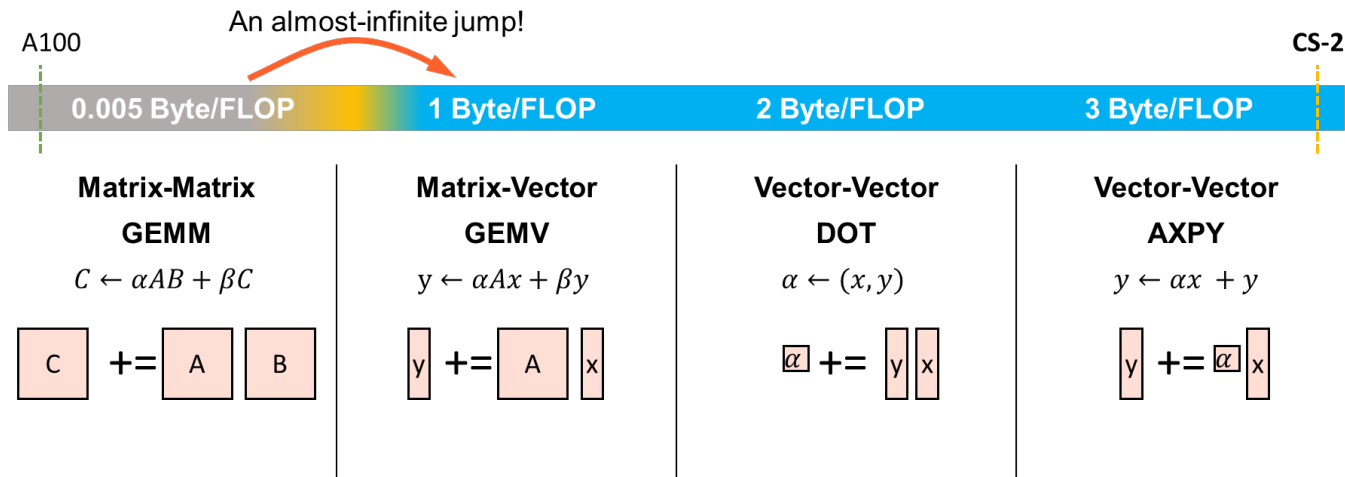
Poll: What hardware do you use today for your HPC work?



# National Energy Technology Laboratory Towards Real-Time CFD

Cerebras system solves sparse linear equations 200X faster than Joule 2.0 supercomputer<sup>1</sup>

Sparse GEMM performance enabled by massive memory bandwidth



1. Rocki et al., "Fast Stencil-Code Computation on a Wafer-Scale Processor" SC20. [arxiv.org/abs/2010.03660](https://arxiv.org/abs/2010.03660)

Photo: Christian Kuhna, CC BY 3.0

# Accelerated energy research at TotalEnergies



**Objective:** Enable order-of-magnitude speedups on a wide range of simulations: batteries, biofuels, wind flows, drillings, and CO2 storage



**Challenge:** Participate in Total study to evaluate hardware architectures, using finite difference seismic modelling code as a benchmark

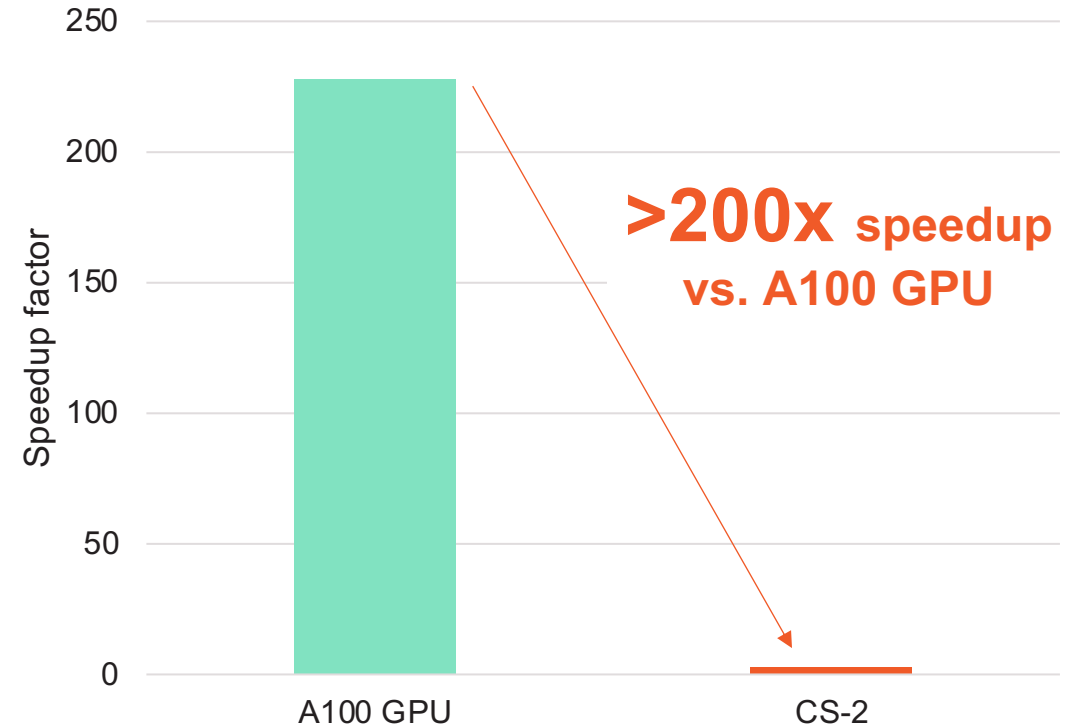


**Outcome:** Cerebras CS-2 system outperformed a A100 AI GPU by >200X using code written in the Cerebras Software Language (CSL). System now installed and running at customer facility in Houston, TX



“We count on the CS-2 system to boost our multi-energy research and give our research ‘athletes’ that extra competitive advantage.”

*Dr. Vincent Saubestre, CEO and President, TotalEnergies Research & Technology USA*



Mathias Jacquelin, Mauricio Araya-Polo, Jie Meng, “Massively Scalable Stencil Algorithm”  
Submitted to SuperComputing 2022, <https://arxiv.org/abs/2204.03775>



# Topics of interest for HPC applications

- Structured grid based PDE and ODE solvers
- Dense linear algebra
- Sparse linear algebra
- Spectral analyses for streaming data
- Particle methods with regular communication
- Monte Carlo type problems that can fill the wafer

# In conclusion

- CS-2 is a dense and powerful single system, powered by 1 enormous chip
  - Cluster-scale compute on a single device => good fit for large DL models
  - 40GB SRAM with massive memory bandwidth => good fit for sparse problems
- CS-2 for Deep Learning
  - Via integration with TensorFlow and PyTorch
  - No low-level programming required
  - Cerebras Software takes care of distributing computations across 850,000 cores
- CS-2 for HPC
  - Via SDK and CSL
  - Users decide how to distribute the computations
- Faster experiments = more ideas tested!



A large, light grey graphic on the left side of the slide, consisting of several concentric, semi-circular arcs that form a partial circle.

# Thank you! Questions?

<https://cerebras.net/>