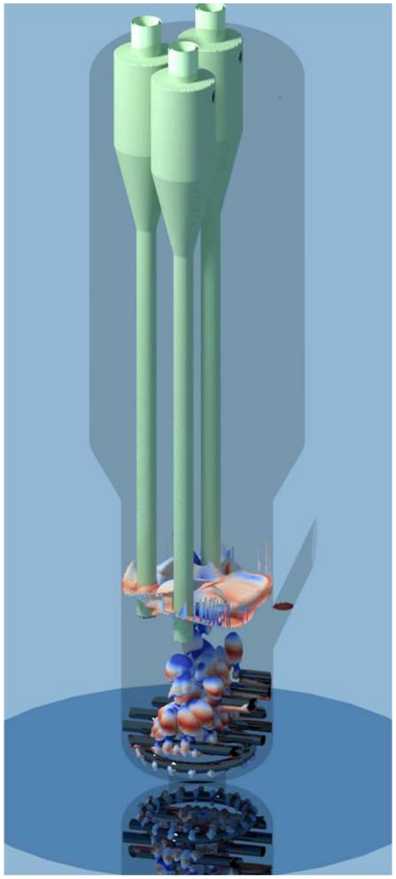
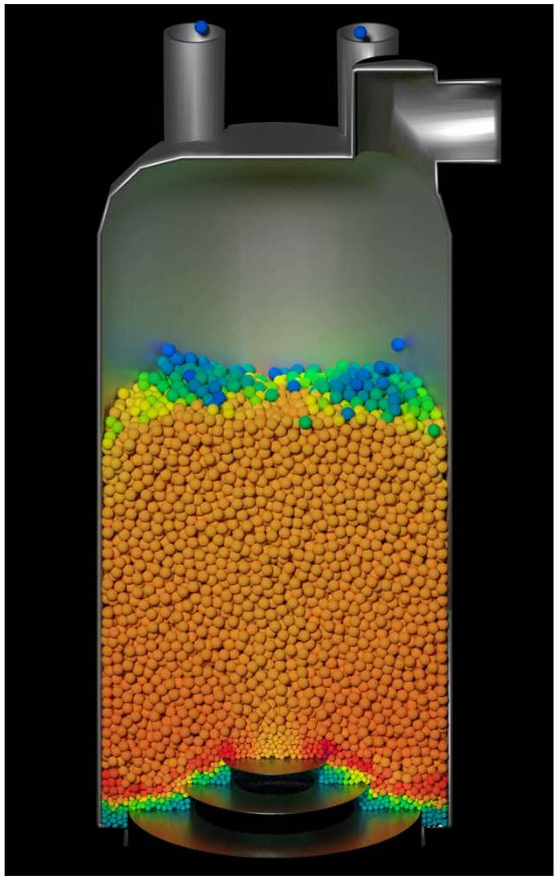
Task 4:Machine Learning to Accelerate CFD Models

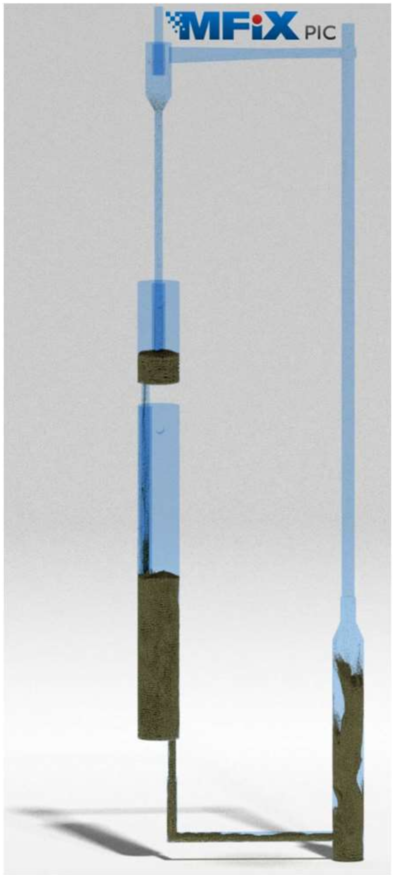
Dirk Van Essendelft, Terry Jordan, Mino Woo, Tarak Nandi

The Need for this Work

Making SBE-CFD Tractable







Project Objective

Research Goal

Build an advanced collaborative framework specifically targeted towards CFD on the most advanced HPC/AI hardware with native support for AI and ML algorithms

Aligned with FE Objectives

Increasing computational speed without sacrificing accuracy will directly supports:

- Modernization of existing coal plants
- development of coal plants of the future
- Reduction of the cost of carbon capture, utilization, and storage (CCUS)

Project Origins

Driving Question

MFiX + TensorFlow

Can we write MFiX in TensorFlow so that we can create a single, unified framework for doing both CFD and AI/ML on emerging hardware designed for AI/ML?

- TensorFlow is the most used AI/ML framework
- TensorFlow has a simple API and allows for both surface level hardware agnostic coding and the ability to deeply optimize hardware specific implementations if needed
- Get speed boosts from AI/ML hardware
- Get speed boost from AI/ML accelerated algorithms
- Simplify implementation of AI/ML models in MFiX

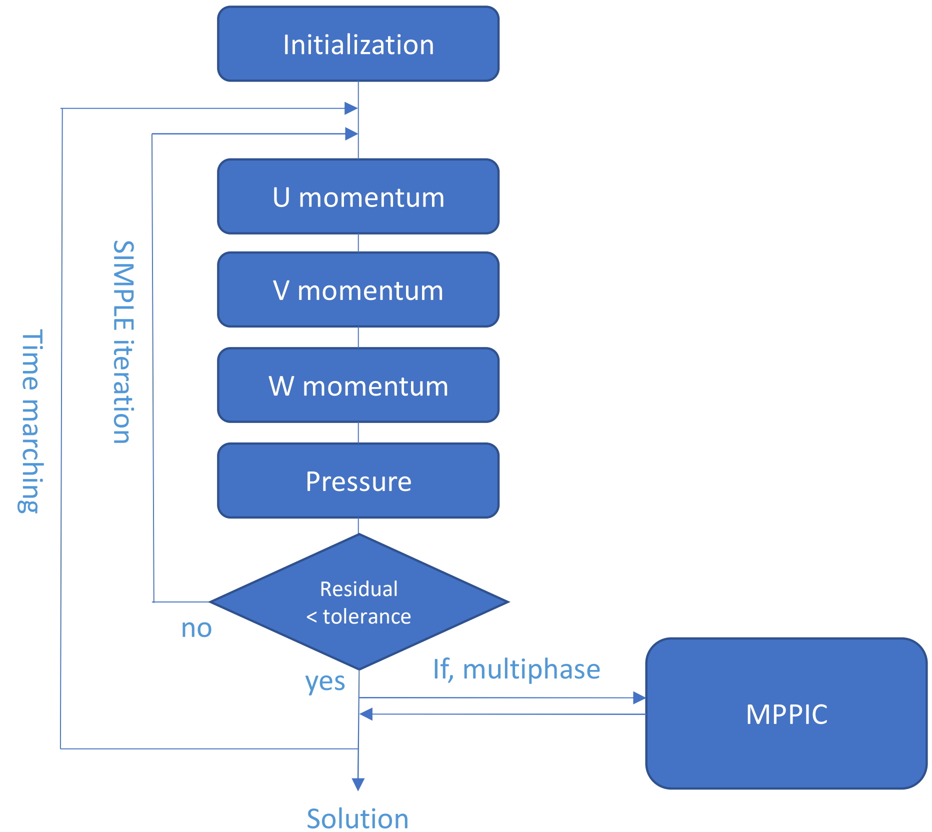
Current Status

Where are we now?

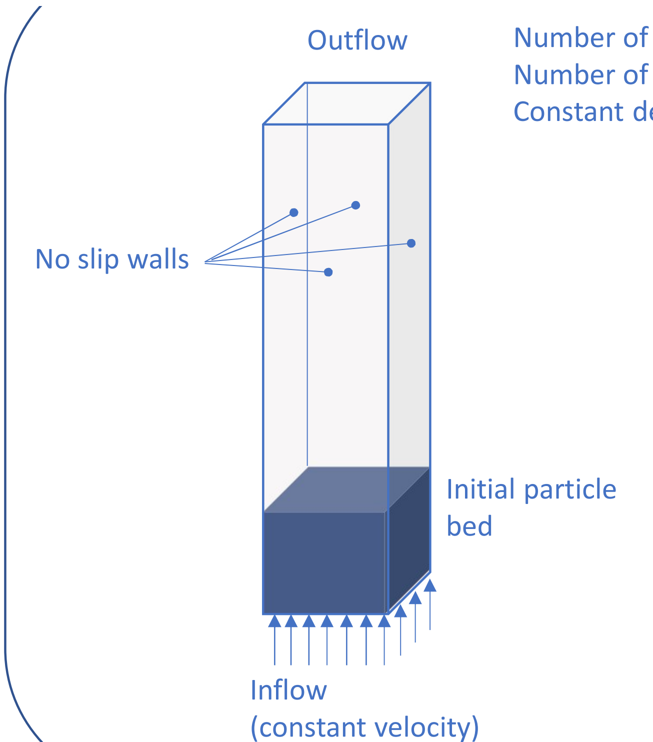
- In Third Full Year of Development, Second year in CARD
- On schedule with Milestones
  - EY20.4.A A demonstration of a multidevice linear solver relative to the existing single device solver (9/30/2020)
  - EY20.4.B A demonstration of a granular simulations using the TensorFlow based solver. (9/30/2020)
  - EY21.4.C A demonstration of a simple fluid bed simulation in the TensorFlow based solver. (3/30/2020)
- Have a functioning, coupled MP-PIC code implemented in TensorFlow
  - Solves all transport equations on available devices followed by a multi-device solve of continuity
  - Ready to accept AI/ML models
  - Does not yet support energy, species, or reactions

Current Status

Verification Against MFiX Classic



Test problem: fluidized bed



Number of cells: 137,924
Number of parcels : 2,983,447
Constant density and viscosity

Current Status

Verification Against MFiX Classic

In first SIMPLE iteration at first time step

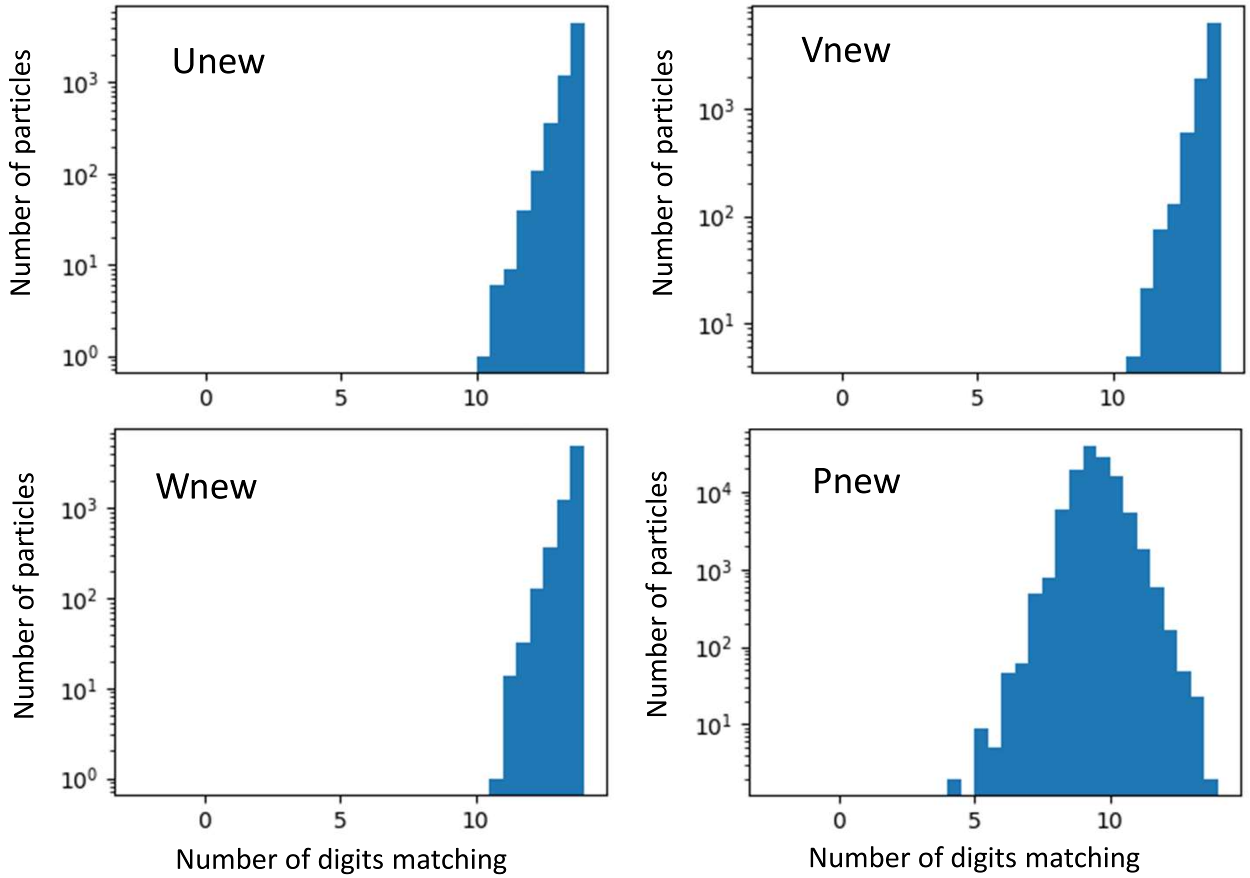
Matching number of digits:

MFiX-Classic: 9.35535931408**4**e-05

MFiX-AI : 9.35535931407**2**e-05

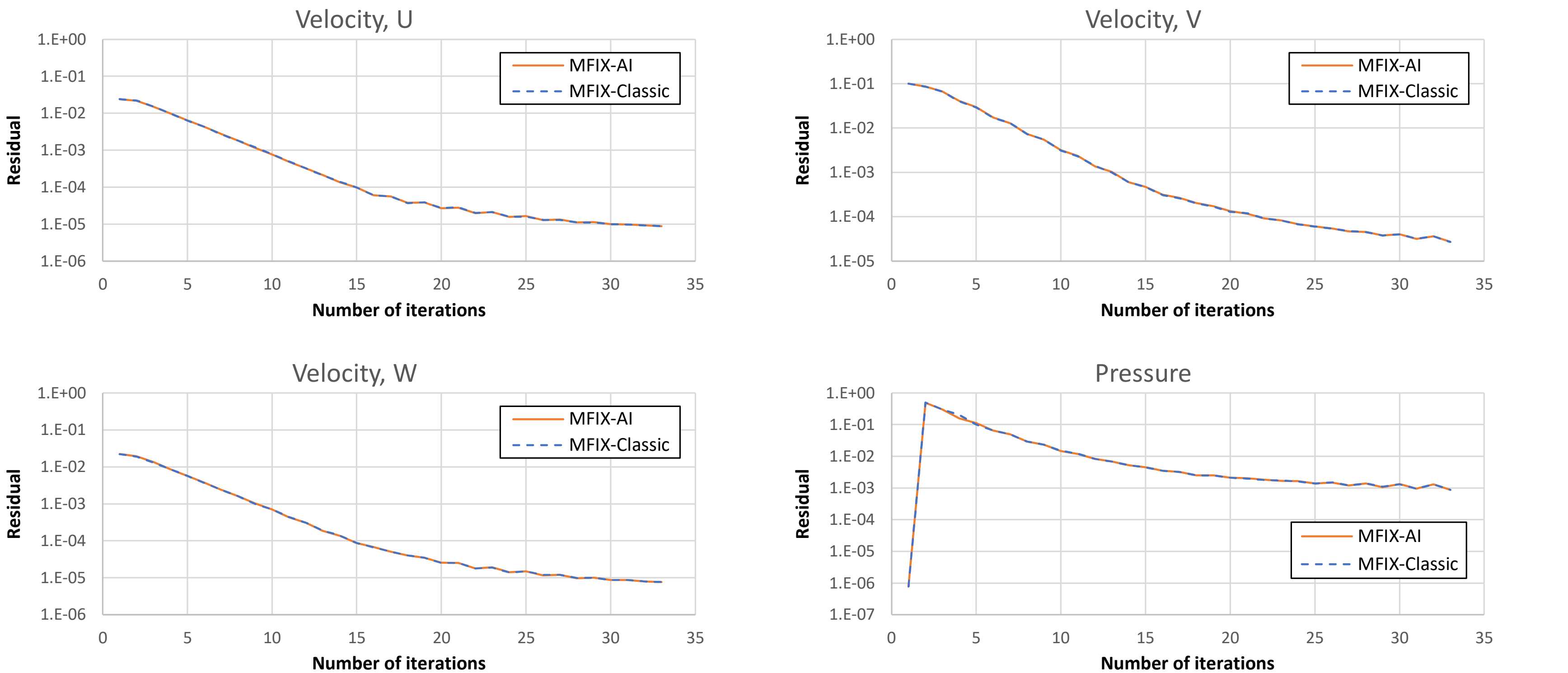
11 digits matching

Provide tighter criteria than relative % error in code-to-code comparison



Current Status

Verification Against MFiX Classic



Current Status

Verification Against MFiX Classic

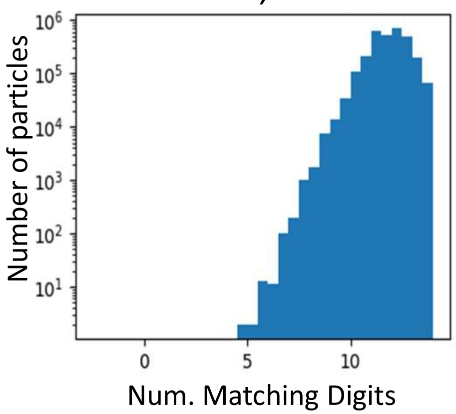
$$\tau_{\mathrm{p}} = \frac{p_{\mathrm{s}} \boxed{\varepsilon_{\mathrm{p}}^{\beta}}}{\max\left[\varepsilon_{\mathrm{cp}} - \varepsilon_{\mathrm{p}}, \alpha(1-\varepsilon_{\mathrm{p}})\right]}$$

Attributed to the divergence between CPUs and GPUs calculation

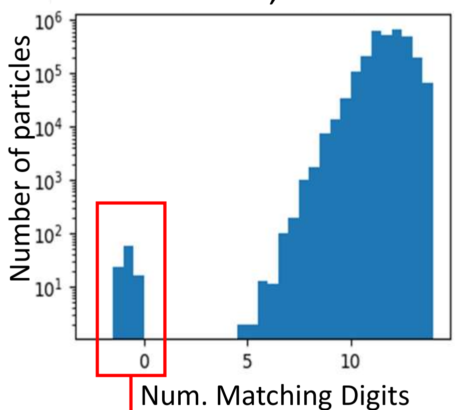
where $\varepsilon_{\mathrm{cp}}$: close pack solid volume fraction

α , β and p_{s} : scalar model parameters

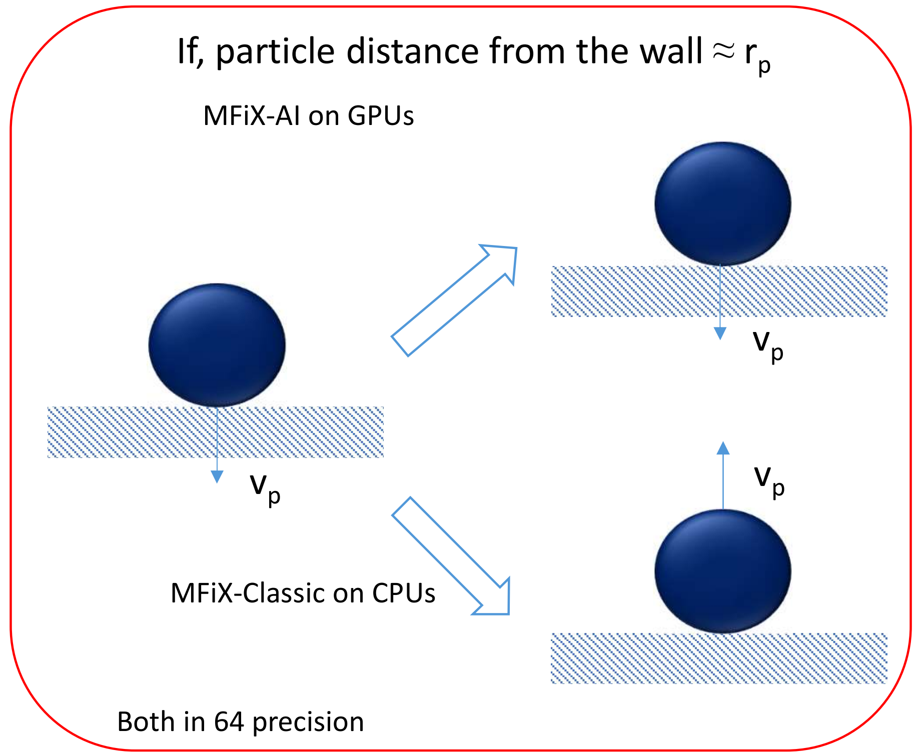
Wsolid, before



Wsolid, after

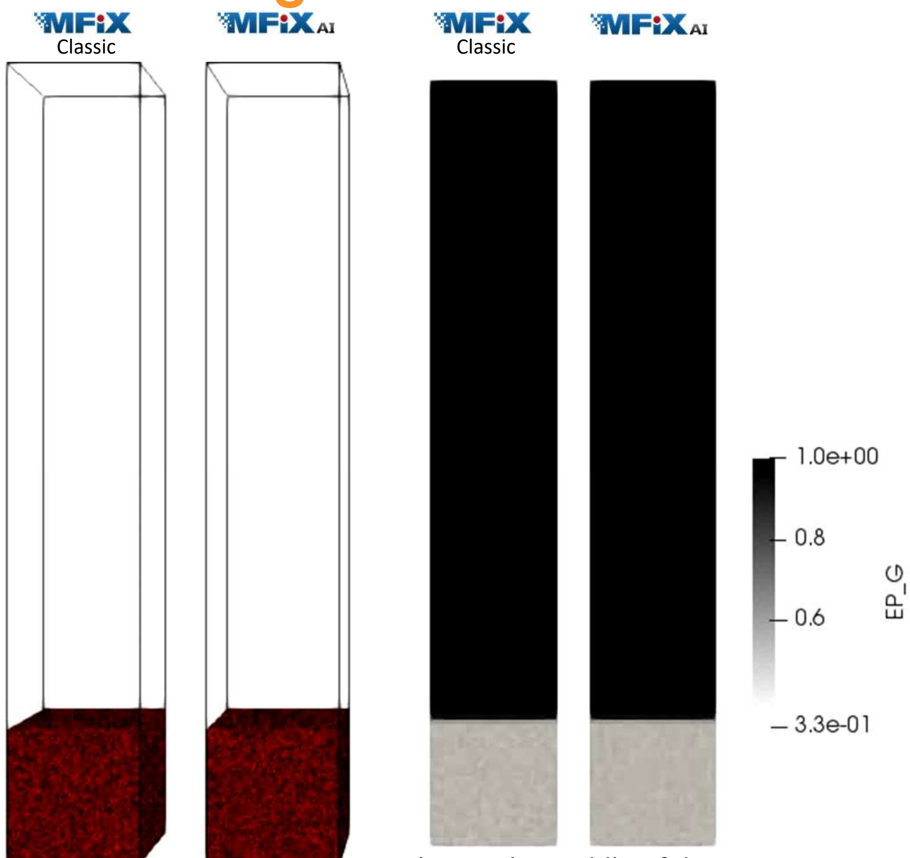


Velocity in opposite direction

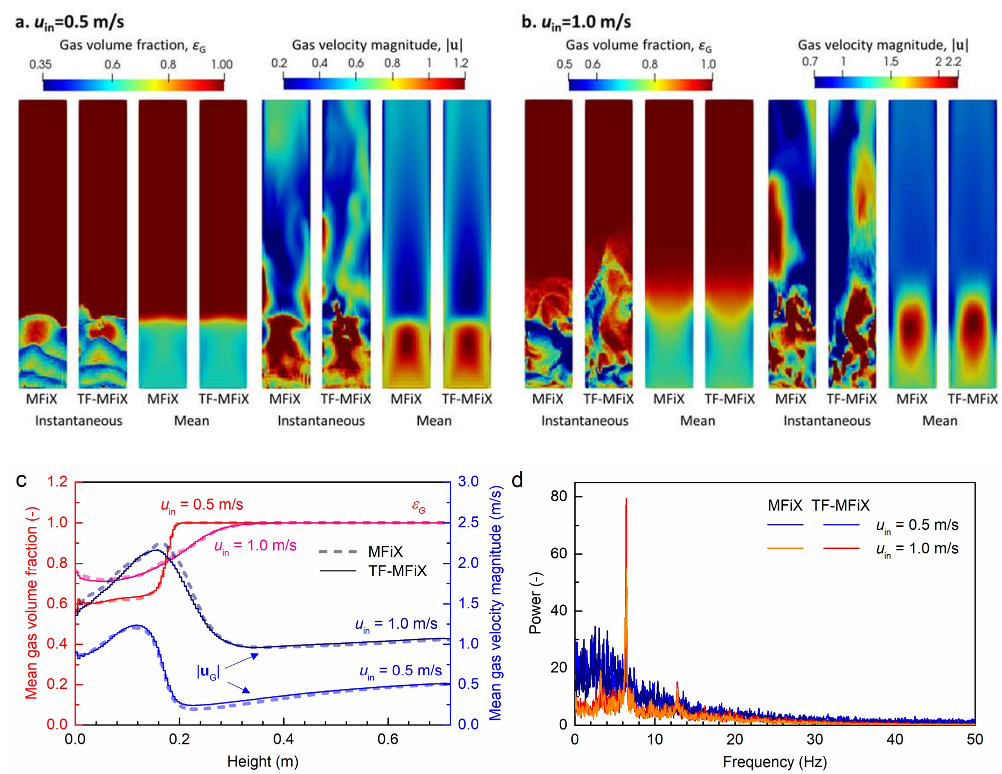


Current Status

Verification Against MFiX Classic

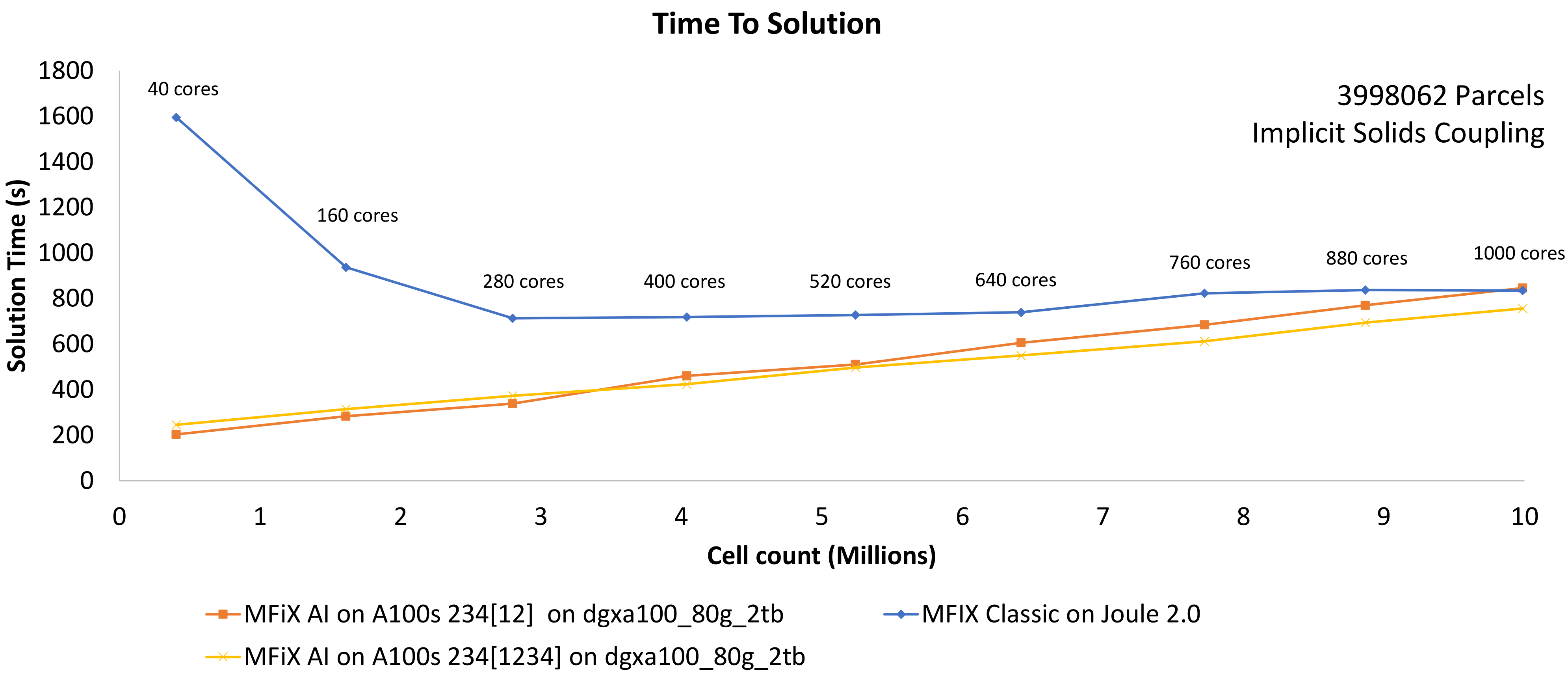


Slice at the middle of domain



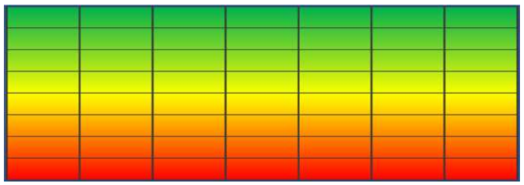
Current Status

Performance Comparison

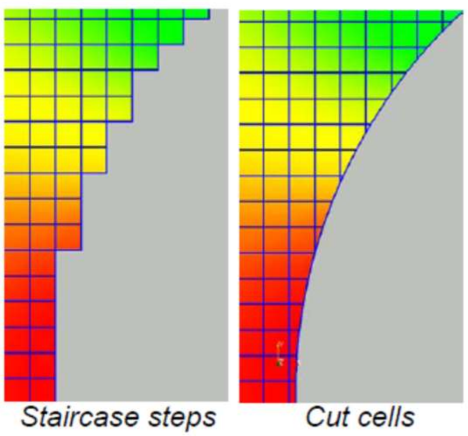


Current Status

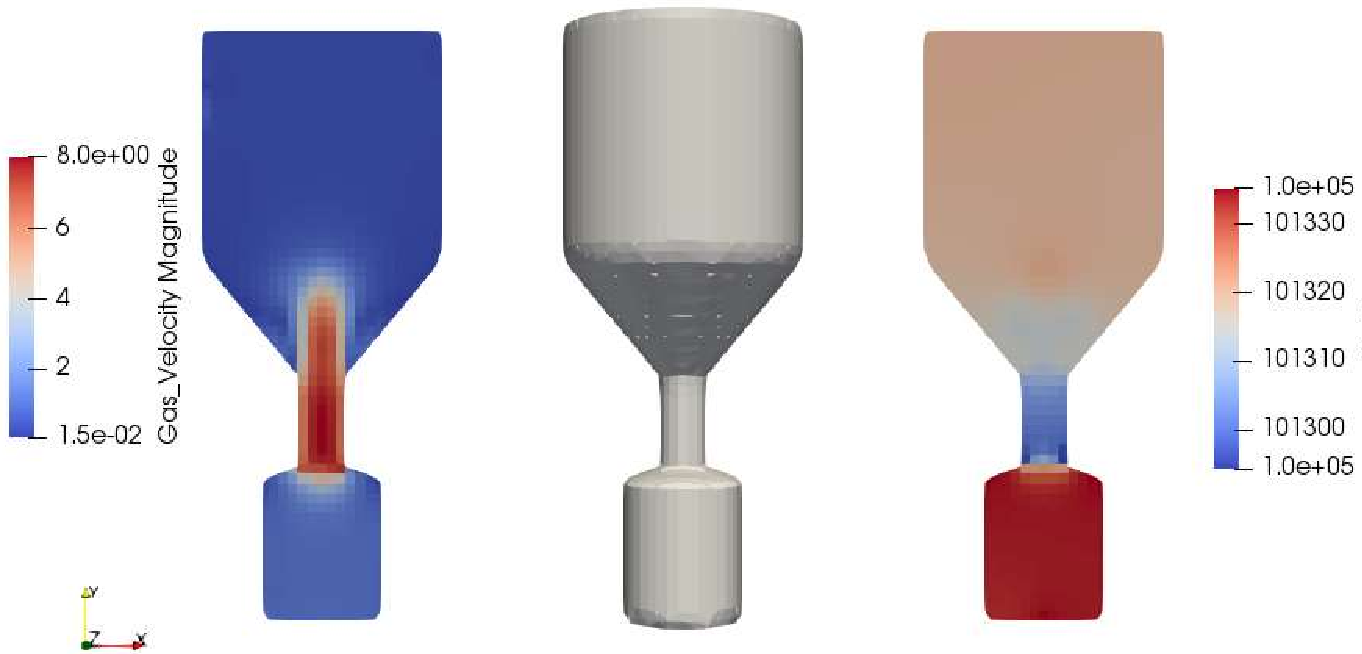
Cut Cell



Uniform grid



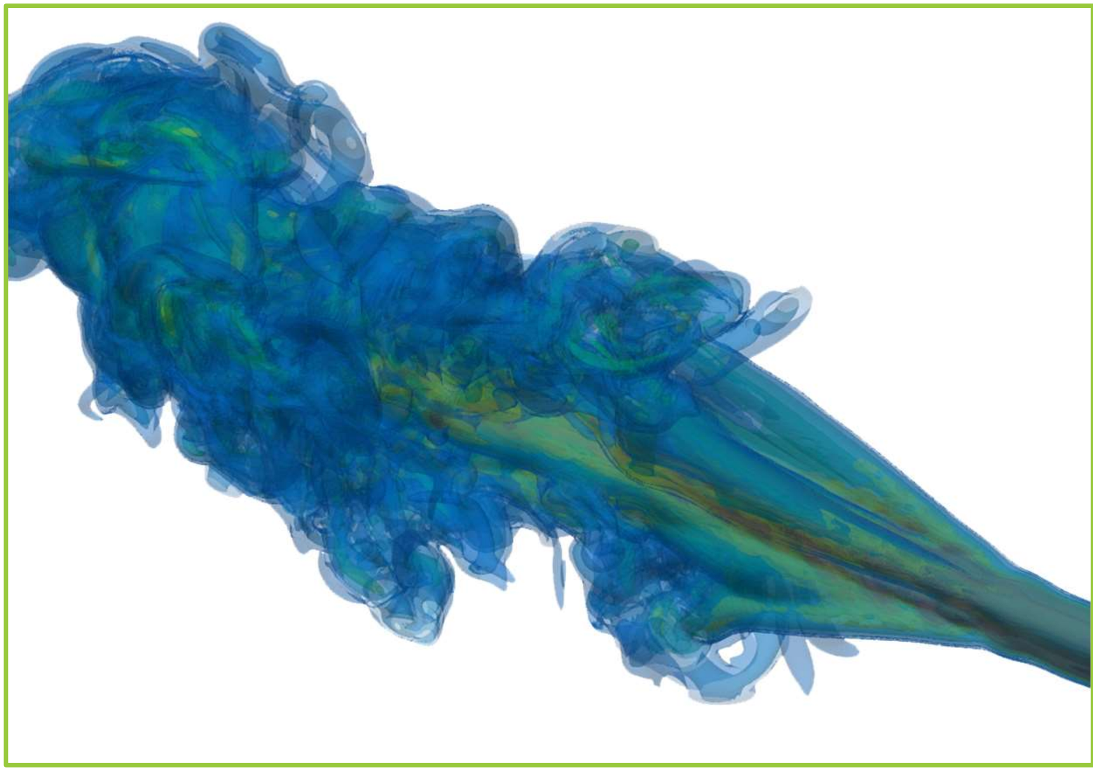
Non-uniform grid*



*Source: Implementation of Cartesian Cut-Cell Technique into the Multiphase Flow Solver MFIX, Jeff Dietiker, April 22, 2009

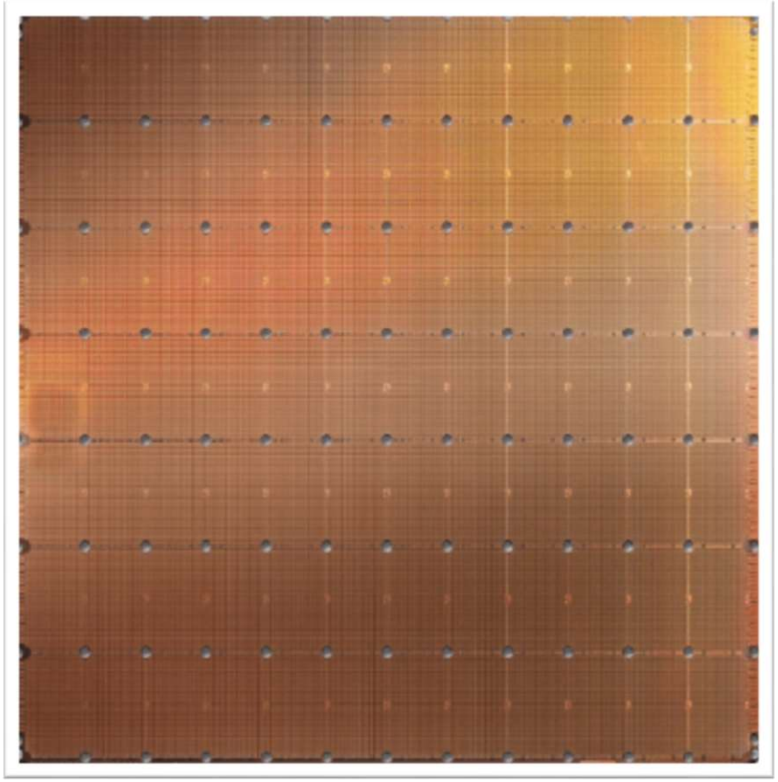
Current Status

Cerebras WSE + MFiX AI



Current Status

Cerebras WSE + MFiX AI

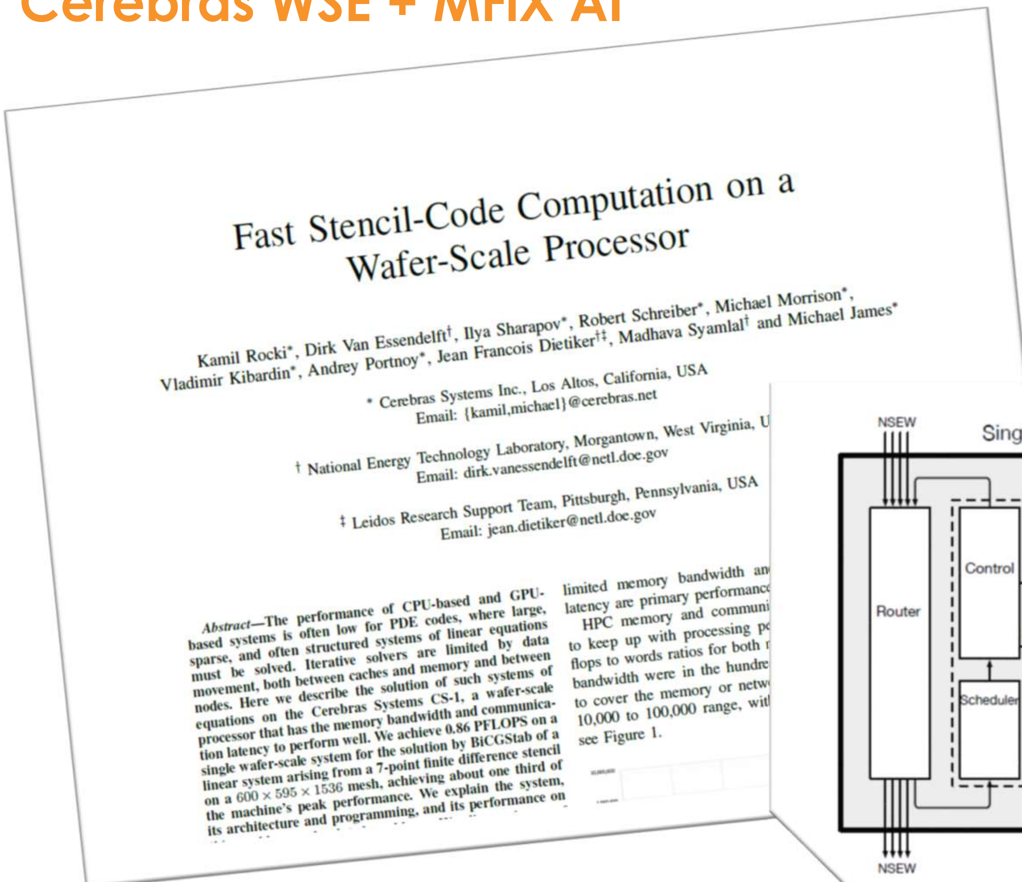


cerebras.net

Size	462 cm^2
Cores	850,000
Transistors	2.6T
Memory Band Width	20 PB/s
Interconnect Bandwidth	220 Pb/s
Memory	40GB
Power	20kW

Current Status

Cerebras WSE + MFiX AI



Fast Stencil-Code Computation on a Wafer-Scale Processor

Kamil Rocki*, Dirk Van Essendelft†, Ilya Sharapov*, Robert Schreiber*, Michael Morrison*, Vladimir Kibardin*, Andrey Portnoy*, Jean Francois Dietiker†‡, Madhava Syamlal† and Michael James*

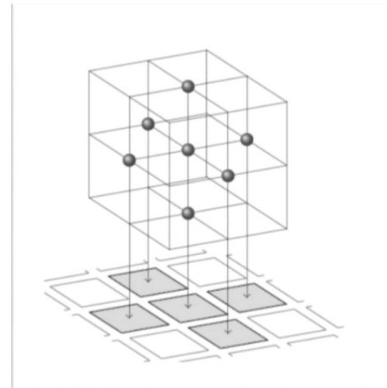
* Cerebras Systems Inc., Los Altos, California, USA
Email: {kamil,michael}@cerebras.net

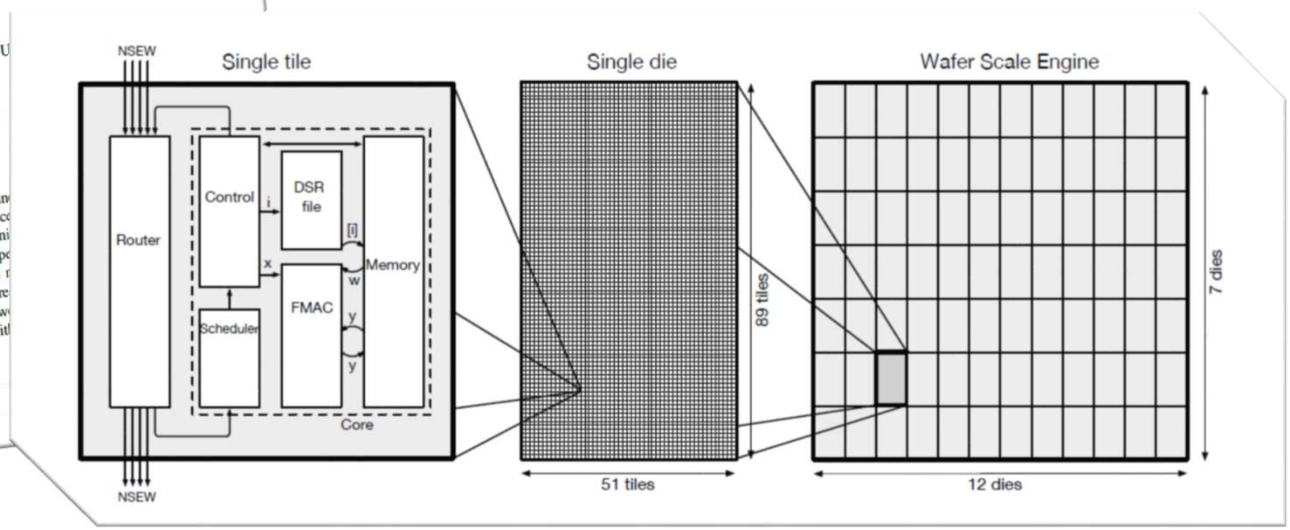
† National Energy Technology Laboratory, Morgantown, West Virginia, U
Email: dirk.vanessendelft@netl.doe.gov

‡ Leidos Research Support Team, Pittsburgh, Pennsylvania, USA
Email: jean.dietiker@netl.doe.gov

Abstract—The performance of CPU-based and GPU-based systems is often low for PDE codes, where large, sparse, and often structured systems of linear equations must be solved. Iterative solvers are limited by data movement, both between caches and memory and between nodes. Here we describe the solution of such systems of equations on the Cerebras Systems CS-1, a wafer-scale processor that has the memory bandwidth and communication latency to perform well. We achieve 0.86 PFLOPS on a single wafer-scale system for the solution by BiCGStab of a linear system arising from a 7-point finite difference stencil on a 600 × 595 × 1536 mesh, achieving about one third of the machine's peak performance. We explain the system, its architecture and programming, and its performance on

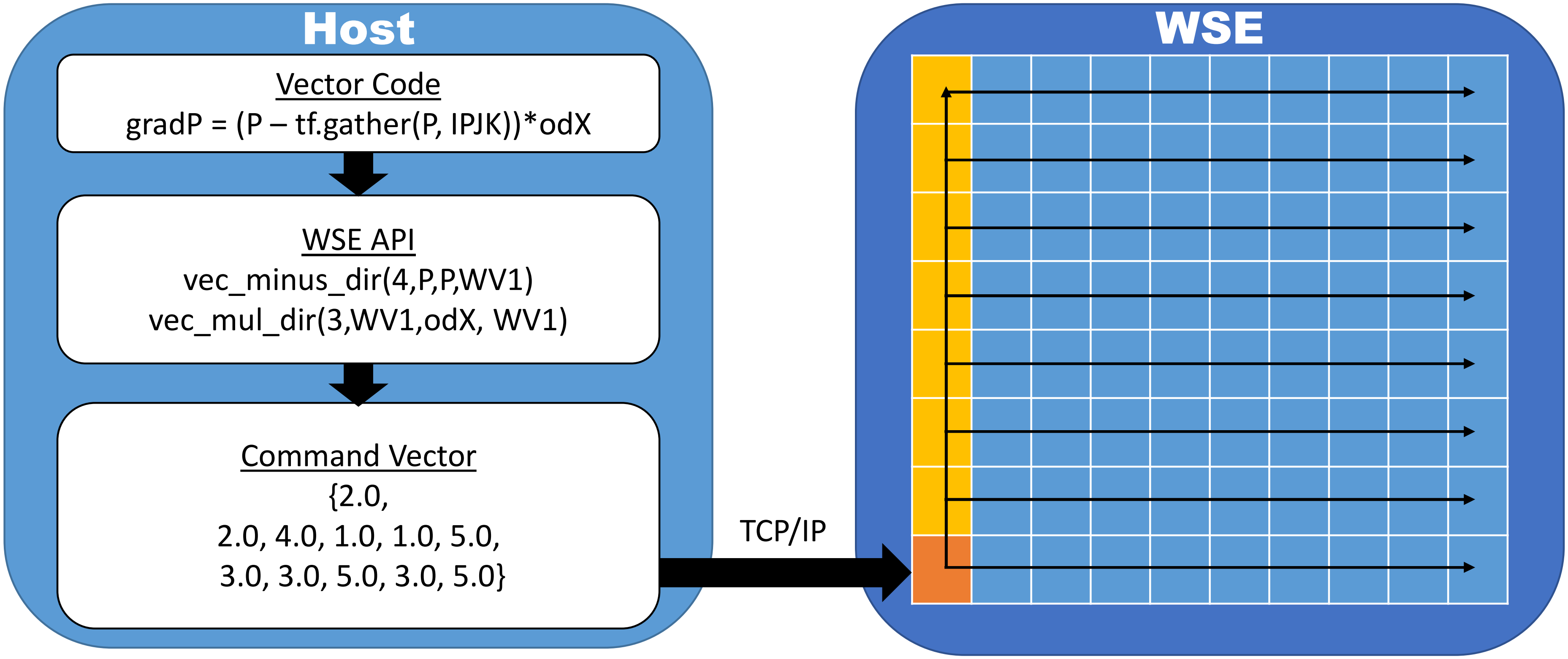
limited memory bandwidth an
latency are primary performance
HPC memory and communi
to keep up with processing p
flops to words ratios for both
bandwidth were in the hundre
to cover the memory or netw
10,000 to 100,000 range, wit
see Figure 1.





Current Status

Cerebras WSE + MFiX AI



Current Status

Cerebras WSE + MFiX AI

First Step Towards CFD: Scalar Diffusion

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

- Explicit method: FTCS (Forward Time/Central Space)

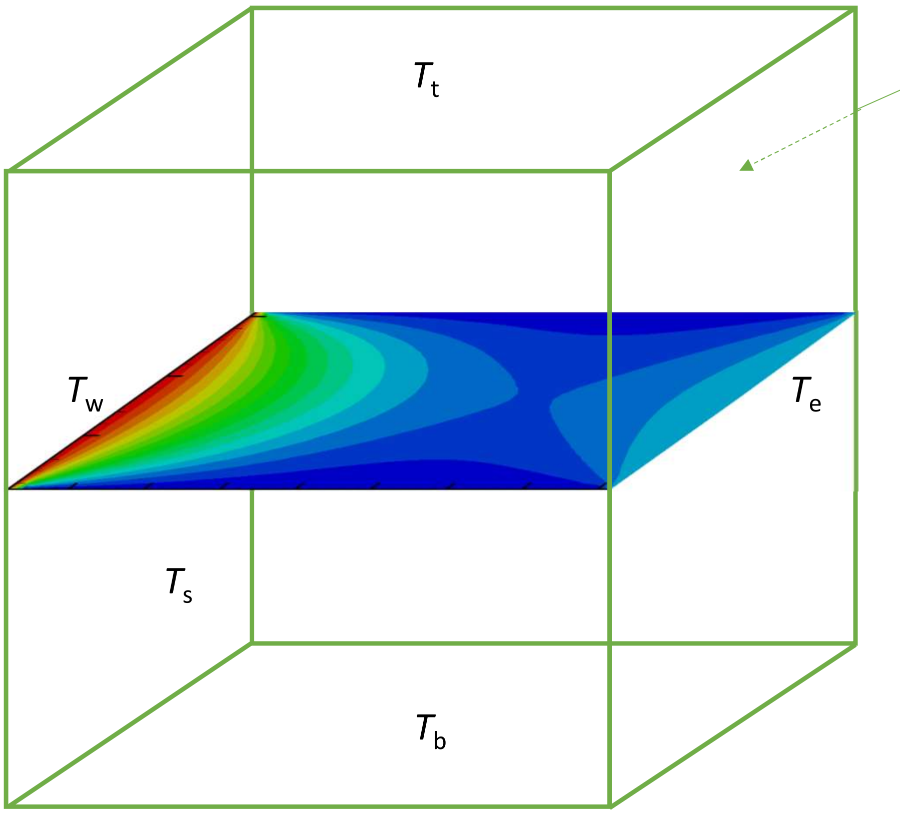
$$T_i^{n+1} = T_i^n + \frac{\alpha(\Delta t)}{(\Delta x)^2}(T_{i+1}^n - 2T_i^n + T_{i-1}^n)$$

Stable for $\frac{\alpha(\Delta t)}{(\Delta x)^2} < 0.5$

- Implicit method: Crank-Nicolson

$$\frac{T_i^{n+1} - T_i^n}{(\Delta t)} = \alpha\left(\frac{1}{2}\right)\left[\frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{(\Delta x)^2} + \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{(\Delta x)^2}\right]$$

Unconditionally sable



Preparing Project for Next Steps

Market Benefits/Assessment

- While CFD is very powerful design tool, the extremely high computational cost limits its practical application
  - 9x improvement on PIC, up to 4x on fluid with current gen hardware with TensorFlow
  - >200x on emerging hardware
  - Makes CFD for optimization and UQ tractable
  - Could open up new application areas in real time or faster than real time CFD

Technology to Market Plan

- Make incremental releases on the existing MFiX platform so existing MFIX users can pick up and run the tools for their FE supported work
- Existing/Potential Collaborators:
  - Industrial relationships with NVIDIA and Cerebras
  - Chris Guenther with TTNEP FWP leveraging framework
  - FOA 2193 - ML models for non-spherical drag
  - Partnering with SAMI as MFiX AI is NETL's first AI enabled CFD code

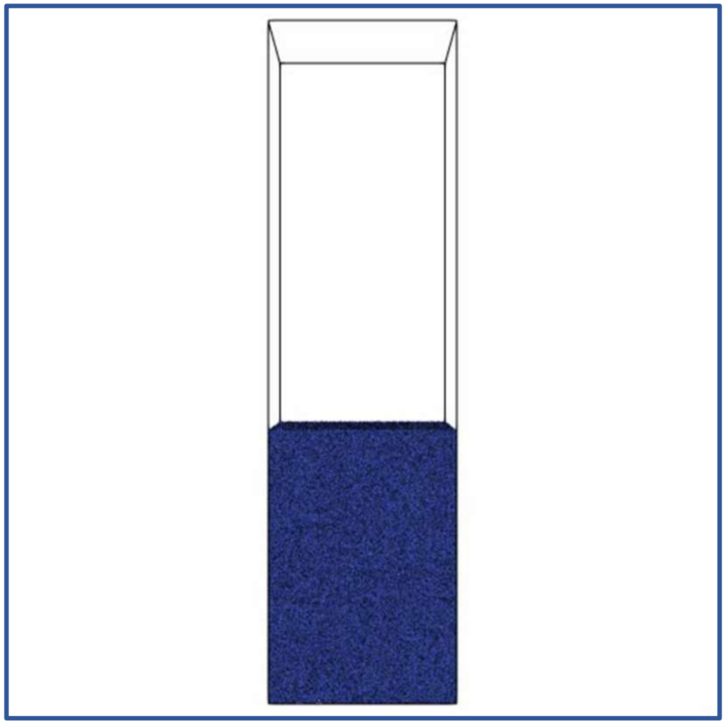
Concluding Remarks

Important Concepts and Next Steps

- The most important thing that this project brings is very high levels of computational speed and efficiency without sacrificing accuracy.
  - Means more work gets done in less time and at lower costs
  - Directly translates to reduced uncertainty, design time, cost, and risk for FE applications
- Project next steps:
  - Funding constraints limited development. Most development is continuing under Sensors and Controls FWP but at a much slower pace
  - A shift to Cerebras Development in FY21 has been brought forward.
    - The goal is to do CFD on the Wafer Scale Engine
    - 200x+ speed gains
    - 1500x+ energy savings
    - First step is to build a minimal linear algebra library that can be applied to solve simple CFD problems
    - Will be linked back to MFiX AI and be released as open source

Current Status

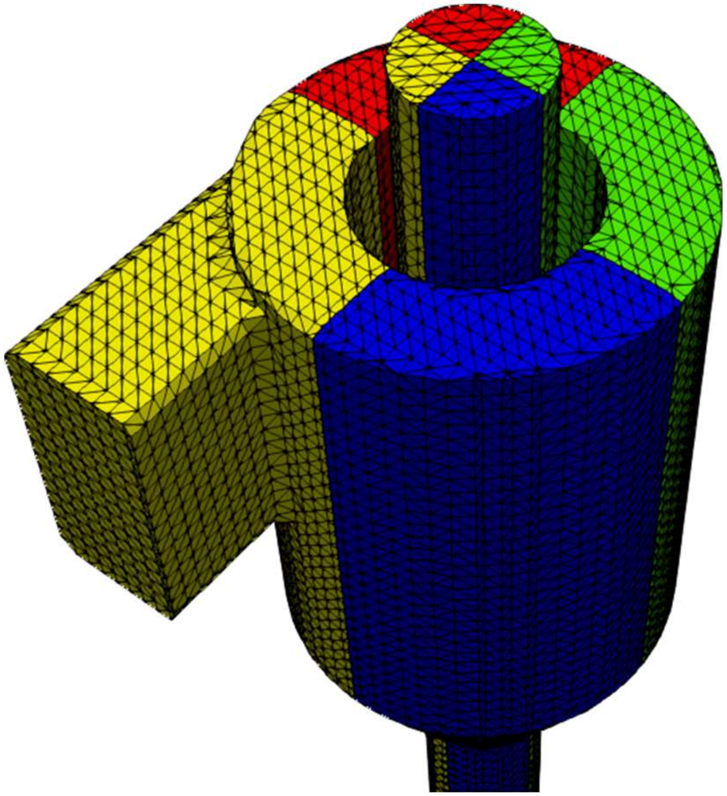
Distributed Computing



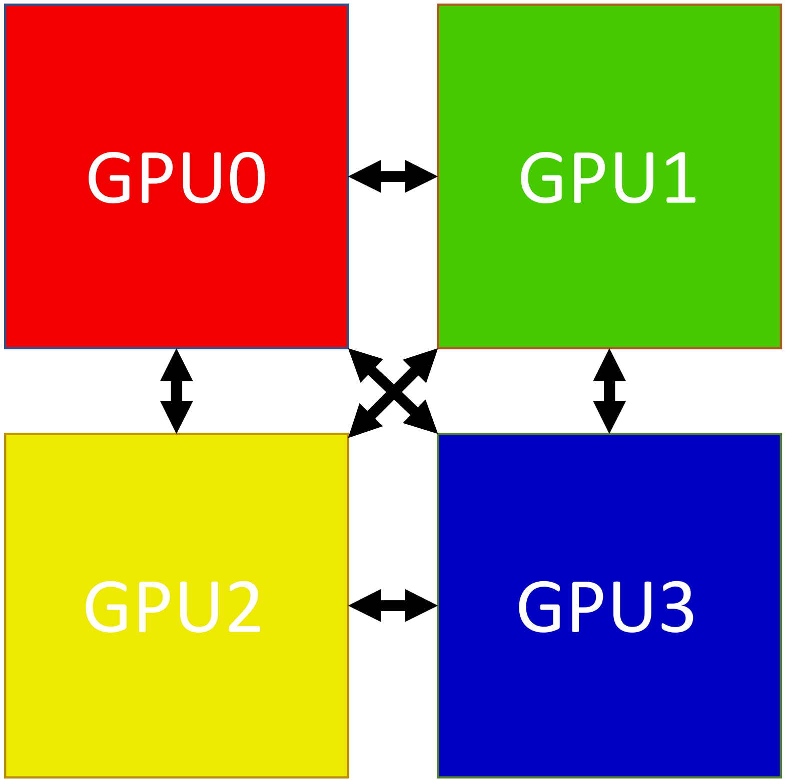
https://developer.nvidia.com/amgx

Current Status

Distributed Computing



Domain Decomposition



Halo Exchange

Current Status

Distributed Computing

AmgX Initialization

(Single, Mutli, Block, Block Global)

- Initialize AmgX and memspace
- Create matrix, vectors, & solver
- Bind and upload init vectors
- Set up the solver

AmgX Solve

- Convert A to CSR
- Update vectors
- Replace the coefficients
- Conversion of A to CSR format
- Perform the solve

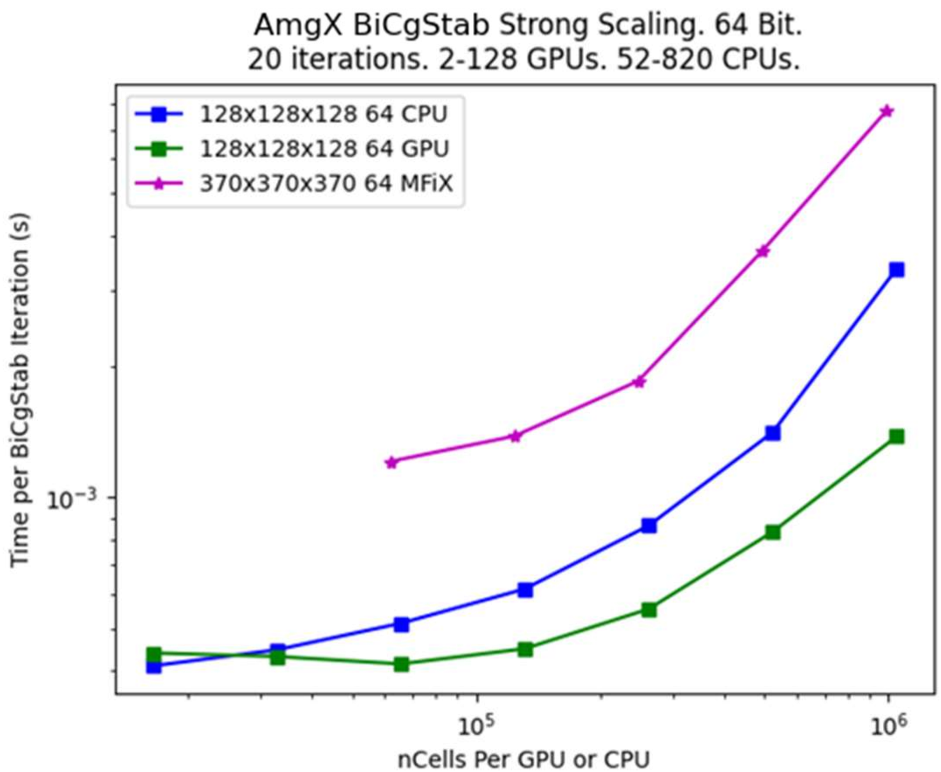
AmgX Cleanup

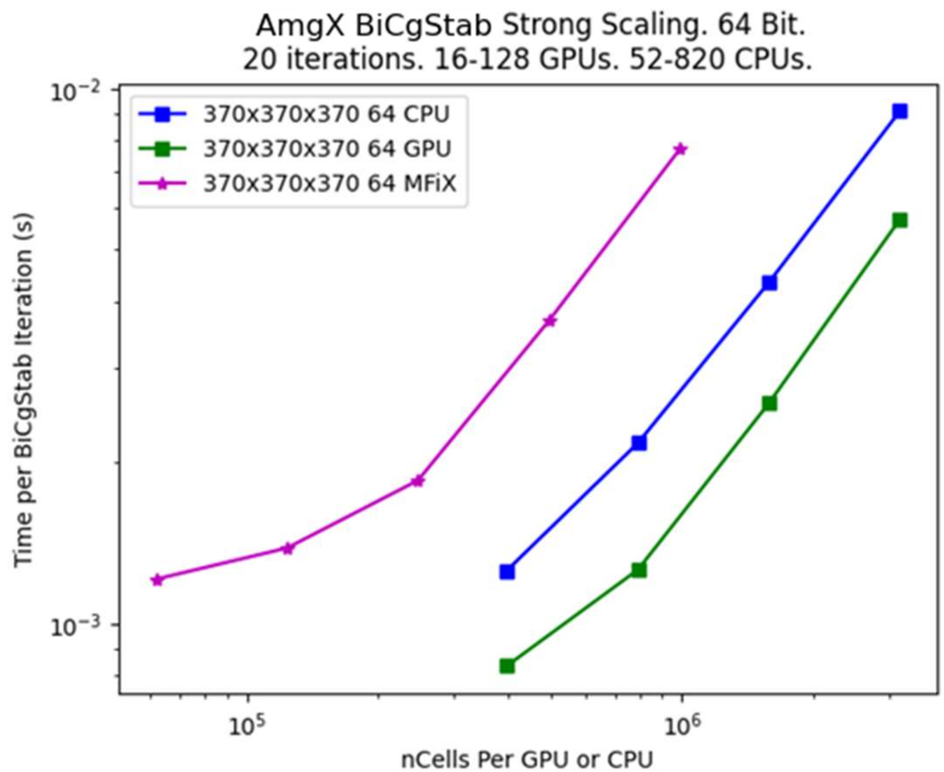
- Destroy matrix, vectors, solver
- Shutdown AmgX
- Free memory

TensorFlow Custom Operators

Current Status

Distributed Computing





Nodes: 2 x Intel Xeon Gold 6148, 2 x NVIDIA P100 PCIe, Intel Omnipath 100Gb