# High Performance Multiphysics Applications in the Post-Exascale Era

**Robert N. Rieben**

With input from: T. Bailey, T. Stitt, D. Hawkins, C. Jekel, J. Belof, D. White, D. Sterbentz, J. Bramwell, K. Korner, W. Schill

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

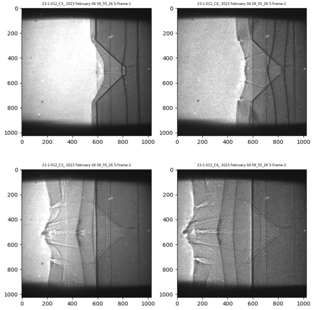
LLNL-PRES-847819

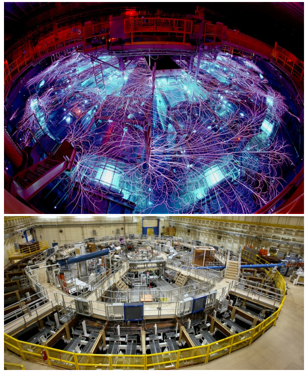
Lawrence Livermore National Laboratory

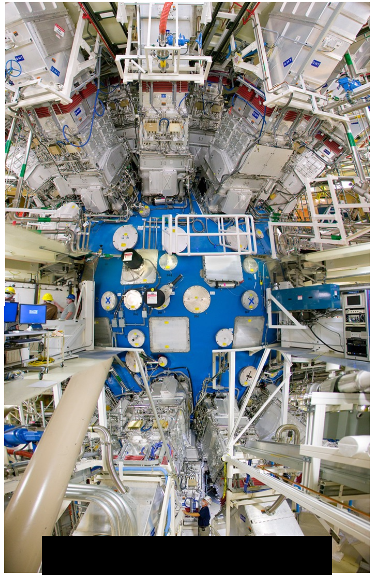
# High performance computing is essential for predictive science at the NNSA

- **High-Performance Multiphysics Simulation**
  - Massively parallel, on-node and off
  - Adaptable, extendable codes
  - Performance portability
  - Multiple architectures and vendors
- **Multiple, Diverse Algorithms**
  - Arbitrary Lagrangian-Eulerian (ALE)
  - Monte-Carlo
  - Discrete Ordinates
  - Smooth Particle
  - Adaptive mesh refinement
  - High-order methods
- **Integrated Applications**
  - Inertial confinement fusion (ICF)
  - Pulsed power experiments
  - Studying material properties in extreme conditions





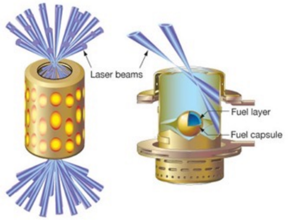



**Our goal is to model and ultimately predict the behavior of complex physical systems**

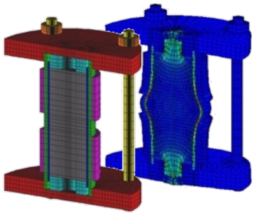
# Our codes are large, complex, tailored to our applications, and represent decades of investment

- Millions of lines of code in multiple programming languages
- Scale to O(1M) MPI ranks
- Multiple spatial/temporal scales
- Maintain connection to prior V&V efforts
- Coordinate with 10–60+ libraries
- 15+ years of development by large teams
- Portable performance
  - Laptops, Workstations, Commodity Clusters, Advanced Architectures (GPUs), Cloud

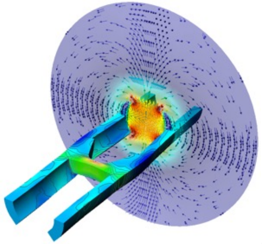
**Inertial Confinement Fusion**




**HE Cookoff**

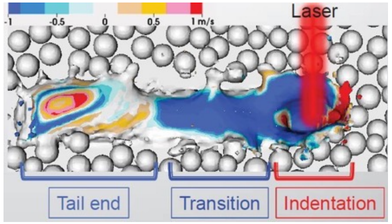


**Navy Railguns**



**Fracture and Failure**

**Additive Manufacturing**



# At LLNL, we have invested in single source, performance portable HPC multiphysics codes over the past 7 years

- Multiple codes, at various levels of maturity
  - Biggest success with C++ codes
- Multiple 3rd party libraries / modules
- Collaboration, organization, regular coordination discussions
- Code projects maintain independence to specialize when necessary, adopt common tools
  - Portability abstraction layers (RAJA suite) essential for success
- Computer Scientists have led the detailed technical R&D for Sierra
- Code Physicists focused development on most impactful application space


NVIDIA

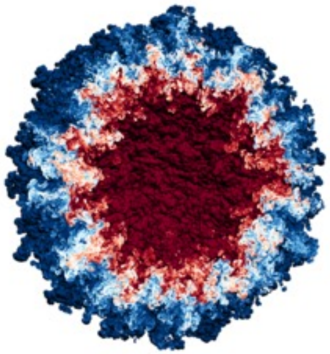
RAJA

Umpire

**Preparing for the Sierra GPU system improved our ability to collaborate across the Lab**

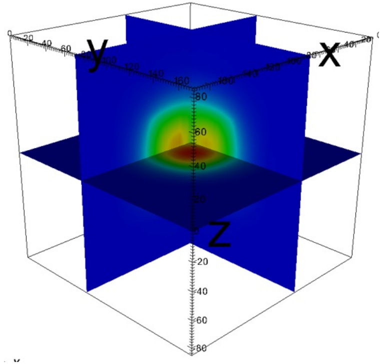
# We have achieved breakthrough performance gains on GPU architectures across multiple codes

**Ares**



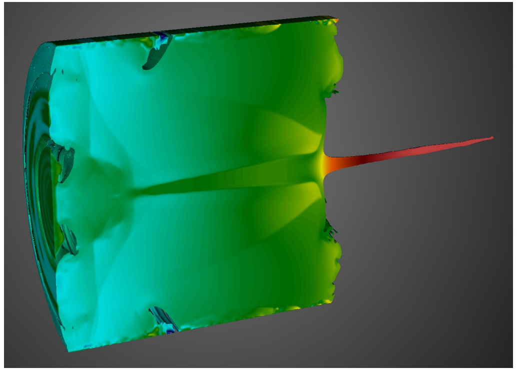
ICF RT Mixing
**13x speedup**

**Ardra**




Reactor Analysis
**16x speedup**

**ALE3D**



Shaped Charge
**8x speedup**

**Goal 1:**
- ✓ Complete today's 3D calculations in a workday

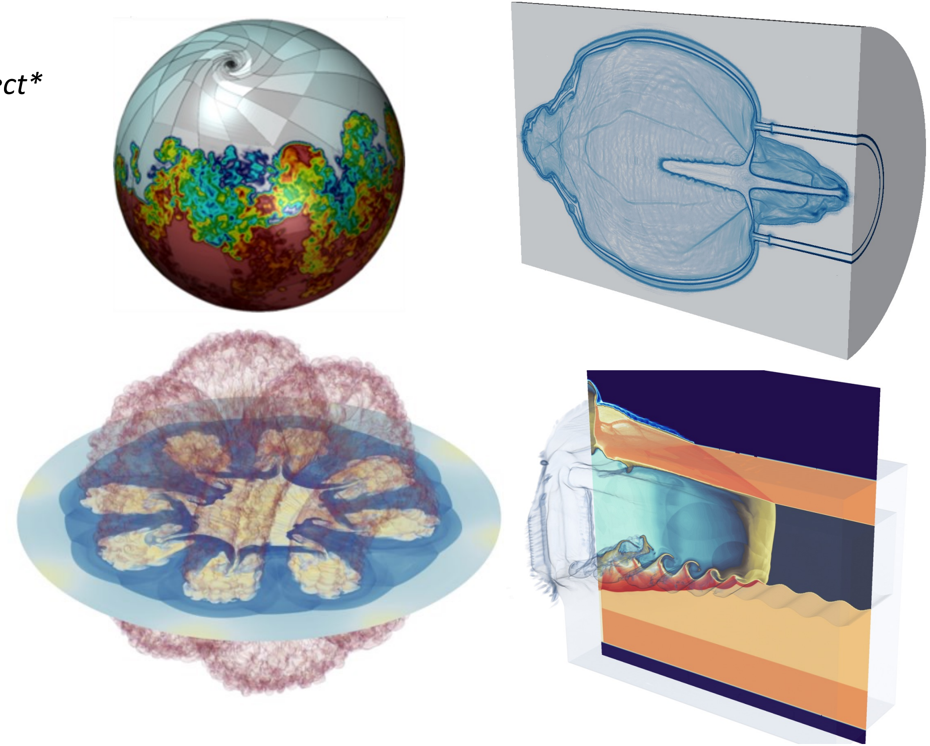
**Goal 2:**
- ✓ Make today's heroic calculations ordinary

**Goal 3:**
- ✓ Establish a new standard for heroic calculations through unprecedented numerical resolution

We intend to capitalize on these gains in the post exascale era

# In addition to refactoring our existing codes, we have stood up a NextGen production code, MARBL

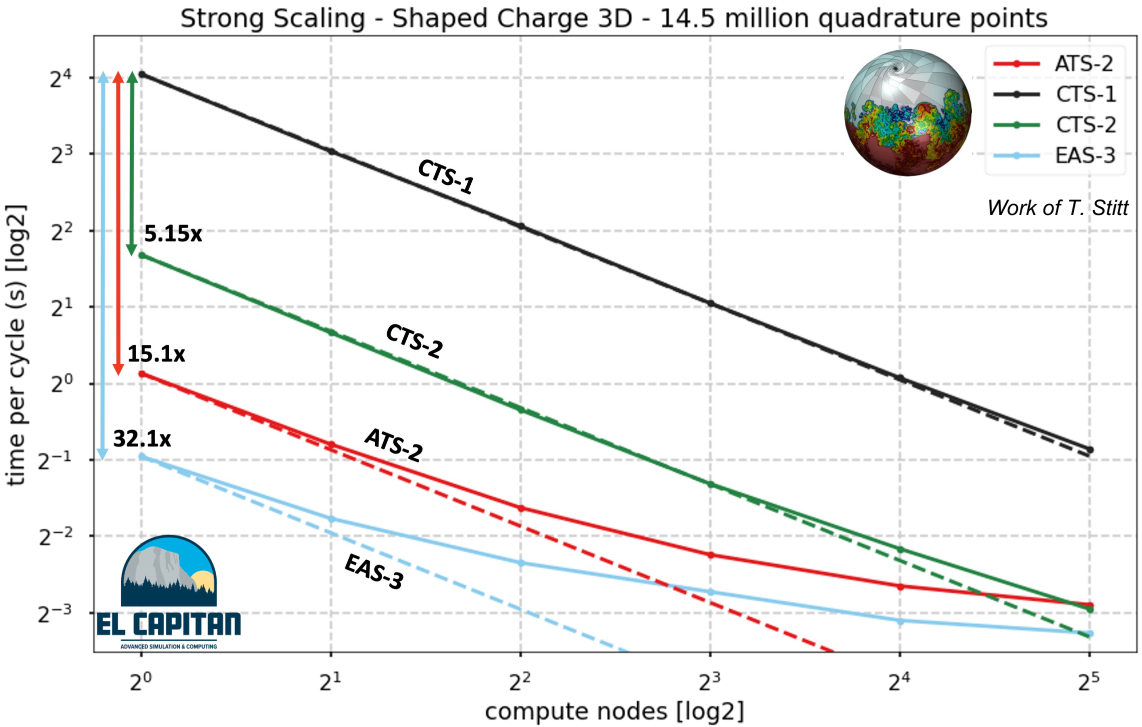
- **Next-gen ICF, pulsed power code**
  - Part of MAPP: *Multiphysics on Advanced Platforms Project**
- **Multiphysics capabilities to date:**
  - Multimaterial compressible ALE hydrodynamics
  - Radiation-hydrodynamics
  - 3T plasma physics + TN burn
  - Resistive MHD
  - RANS turbulence models
- **High-order numerical methods**
  - Higher FLOP/byte, improved GPU throughput
- **GPU performance**
  - >15X GPU vs CPU node speedup



**R. Rieben, K. Weiss et. al., "The Multiphysics on Advanced Platforms Project. LLNL report," LLNL-TR-815869, December 2020.*

# As part of the ASC ATDM program, portable performance on advanced architectures has been a key focus for MARBL

- ✓ Scaled to half of Sierra (2048 nodes)
  - IBM Power9 CPU + NVIDIA V100 GPU
- ✓ Scaled to all of Astra (2496 nodes)
  - Cavium Thunder X2 CPU (ARM)
- ✓ Scaled to most of EAS3 (32 nodes)
  - AMD EPYC CPU + MI250X GPU
  - Early access El Capitan system
- >15x GPU speedup compared to LLNL's "commodity" (CTS) hardware
- Speedups are measured ***node-to-node***

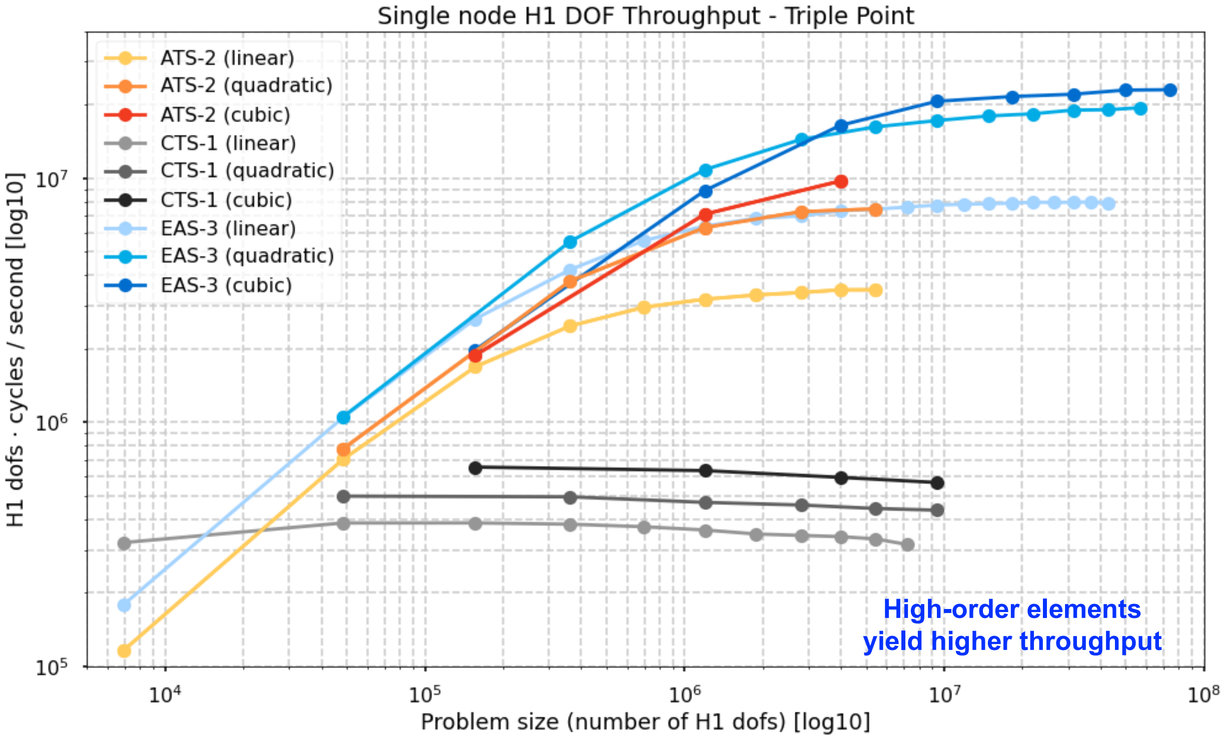



Umpire

See: A. Vargas et. al, "*Matrix-free approaches for GPU acceleration of a high-order finite element hydrodynamics application using MFEM, Umpire, and RAJA,* " Int. J. HPC Apps.

# Algorithms matter: partially assembled “matrix free” methods perform better at high-order on GPUs

- MARBL uses “partial assembly” for high-order finite element operators*
- No full assembly of mass/stiffness matrices
- Instead, compute operator action on vectors
- Has minimal data motion
- Improved performance at high order, especially in 3D on GPUs

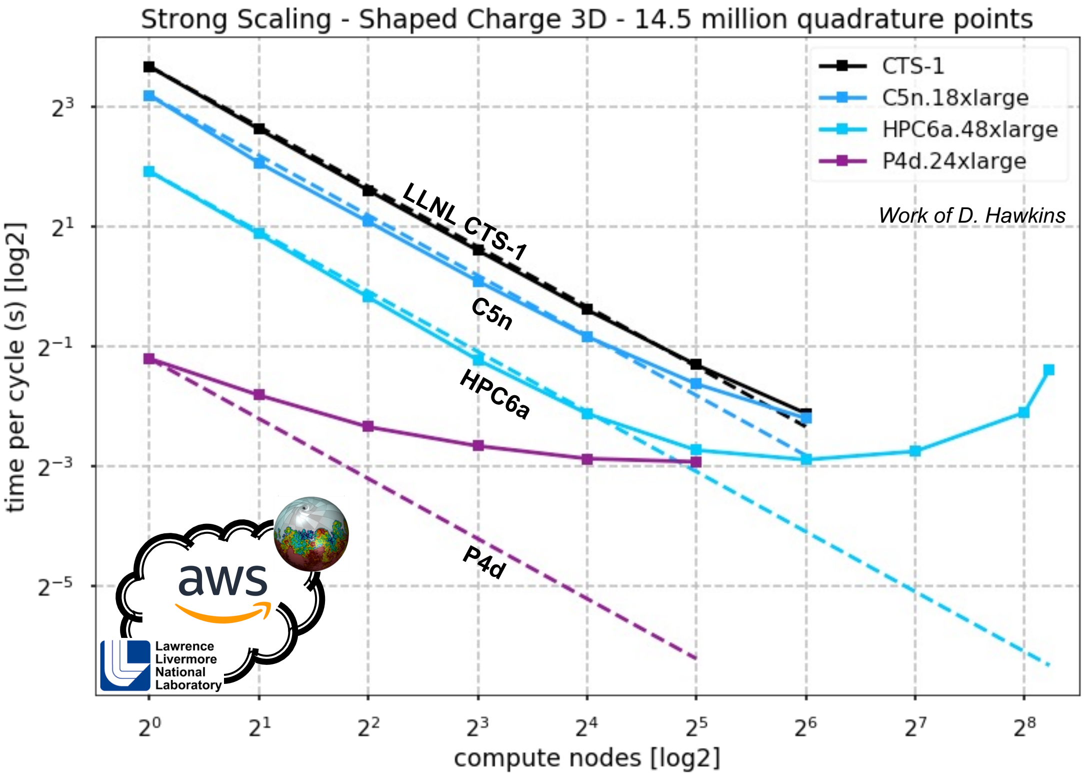




*See: A. Vargas et. al, “*Matrix-free approaches for GPU acceleration of a high-order finite element hydrodynamics application using MFEM, Umpire, and RAJA,* ” Int. J. HPC Apps.

# We have been exploring use of cloud compute resources for our multiphysics applications

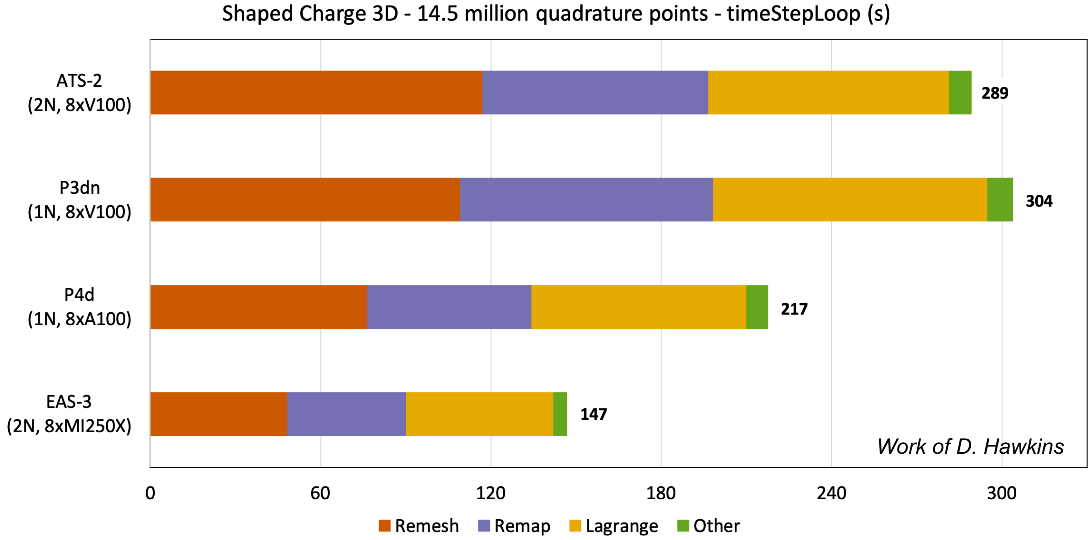
- We are exploring cloud HPC as part of an MOU between LLNL and Amazon AWS
- Containerized MARBL code has no performance degradation compared to standard binary
- We have scaled containerized MARBL on AWS out to 300 nodes (28,800 cores)

| Cluster | CPU | GPU | Interconnect |
|---|---|---|---|
| AWS C5n.18xlarge | 2x18 core Intel Skylake | - | AWS EFA |
| AWS HPC6a.48xlarge | 2x48 core AMD Milan | - | |
| AWS P4d.24xlarge | 2x24 core Intel Cascade Lake | 8xNVIDIA A100-40GB | |
| LLNL CTS-1 | 2x18 core Intel Broadwell | - | Cornelis Networks Omni-Path |



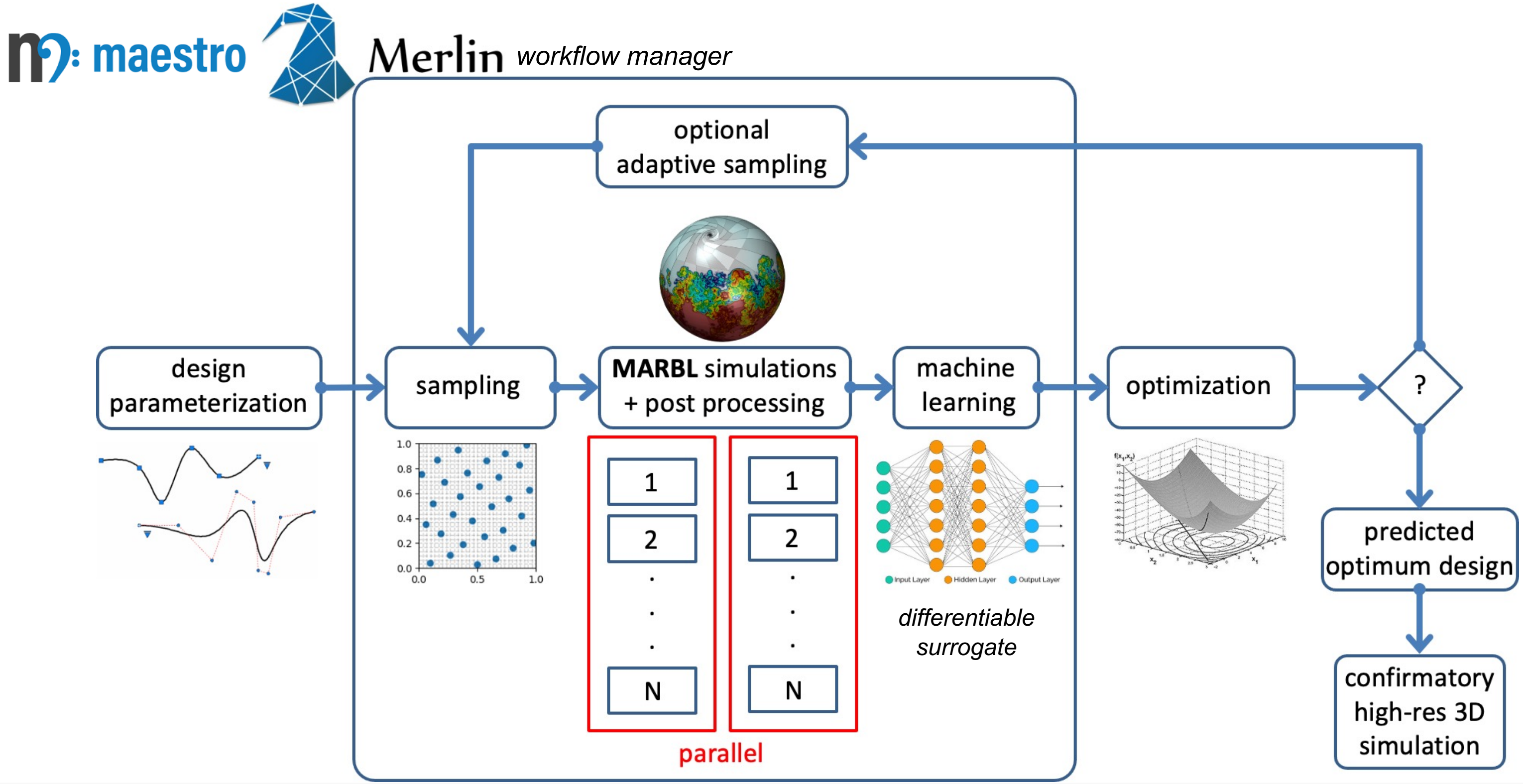
# There are several appealing aspects of utilizing cloud compute in combination with our on-premise HPC

- Containerized code can readily be deployed in GovCloud for certain workloads
- We can easily explore alternate node configurations
- User workflows can be customized by compute needs
- Cloud tools for ML can be leveraged

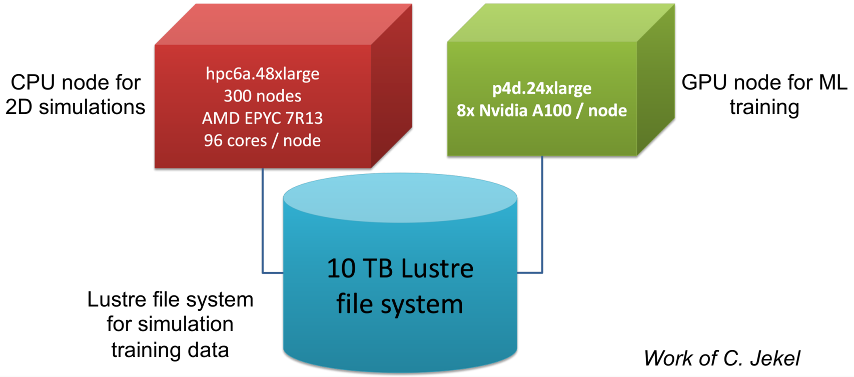

| Node Type | CPU | Memory | GPU |
|---|---|---|---|
| LLNL ATS-2 | IBM POWER9 (2x22 core) | 256GB | 4 x NVIDIA V100-16GB |
| LLNL EAS-3 | AMD Trento (1x64 core) | 512GB | 4 x AMD MI250X-128GB |
| AWS P3dn.24xlarge | Intel Skylake (2x24 core) | 768GB | 8 x NVIDIA V100-32GB |
| AWS P4d.24xlarge | Intel Cascade Lake (2x24 core) | 1152GB | 8 x NVIDIA A100-40GB |

# We are combining MARBL simulations with machine learning to enable optimization driven workflows

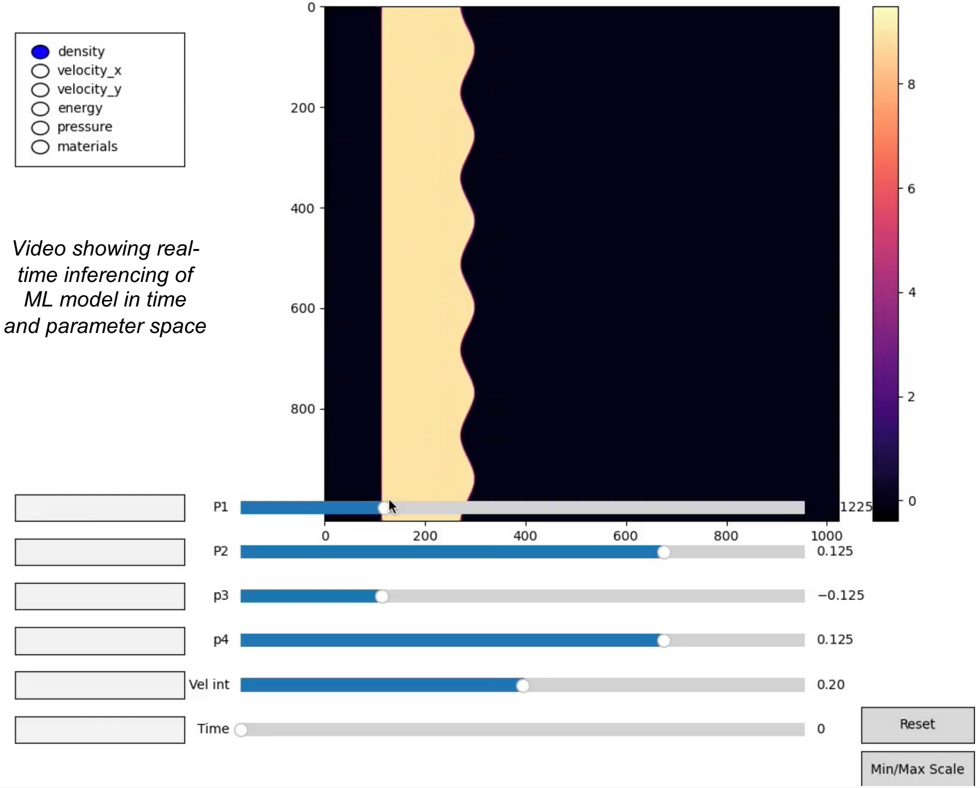


# Cloud computing is well suited for 2D ML ensemble studies where simulation throughput is important

- High velocity impact target optimization
- ML model trained on ~3K MARBL simulations
  - 600 MB data/simulation, extracted using Ascent
  - 51 time snapshots/simulation
- Inputs:
  - 4 spline parameters, flyer velocity and time
- This model was trained in AWS GovCloud
  - 300 nodes, 3K simulations done <1hour



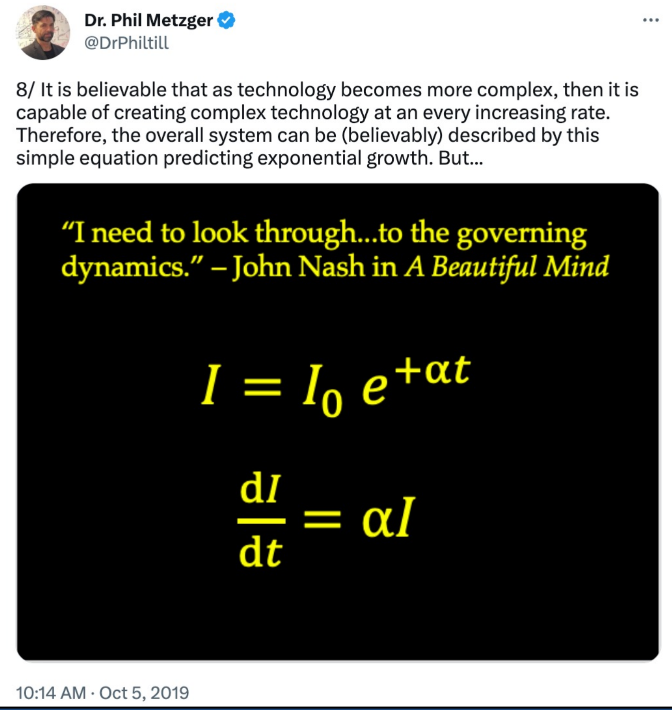

*Work of C. Jekel*




*Video showing real-time inferencing of ML model in time and parameter space*

# Our pursuit of computational performance is not stopping

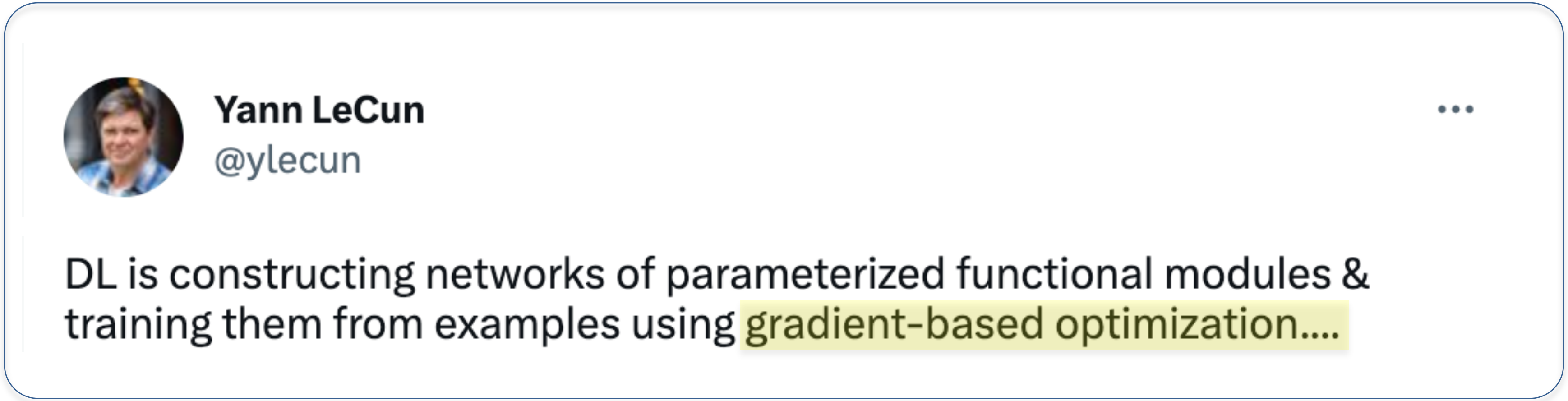
- Hardware performance may be stagnating
- However, Moore's law can be interpreted in a more abstract way
  - Exponential growth in compute come from using current tech generation to accelerate development of next generation
- Keys to continued performance:
  - Build on success of previous technology cycle
  - Verified/validated multiphysics codes become a building block in a bigger piece of software
  - Every calculation will be an ensemble




**We see ample opportunity for more performance gains coming from software and algorithms**

# We believe the key to unlocking future performance gains will come from gradient based optimization

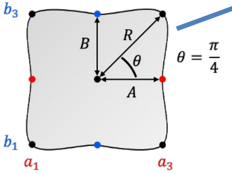
- Purely data driven approaches are not a viable substitute for our simulation codes
- Experimental data is expensive to generate and insufficient to cover our uncertainties
- We often use our codes to **produce training data** for surrogate models
- However, there is much to gain from the world of AI / ML …

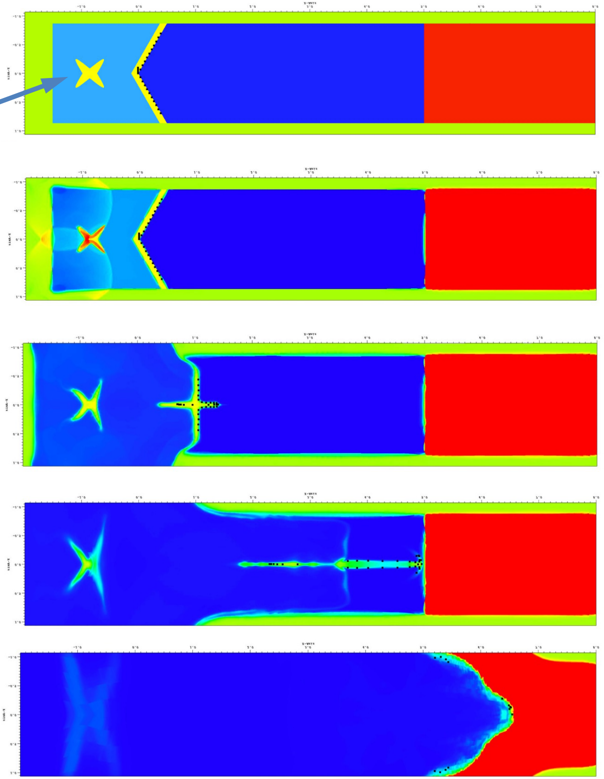


# Multiphysics codes combined with ML models can be used to explore high-dimensional design space

- For a planar shaped charge, we can optimize deflector geometry to maximize jet penetration
- The ML model maps design parameters to penetration depth
- Model training requires *O(10k)* MARBL simulations

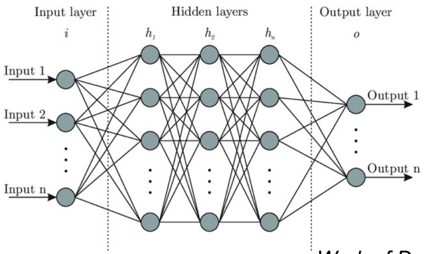
deflector shape parameterization



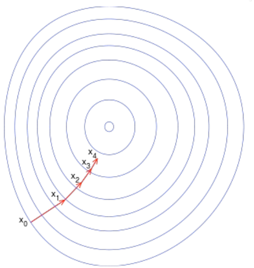



simulation time

ML Surrogate Trained on Data




Gradient-Descent on Surrogate

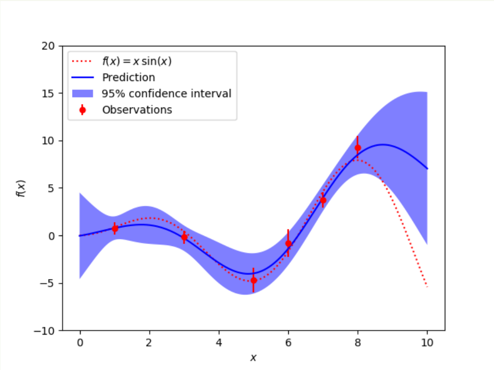



*Work of D. Sterbentz, D. White*

# Gradient-enabled multiphysics simulation codes will allow much larger design space explorations

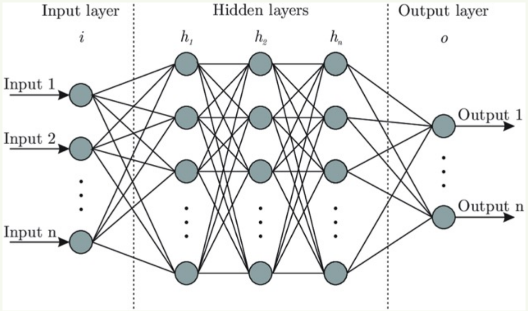
## Black Box (Gradient-Free)

- ✓ Can use existing simulation tools
- ✓ Non-intrusive for code
- ✓ Good for exploration
- • Requires many simulation samples
- • Limited to O(10) design parameters




## ML Surrogate

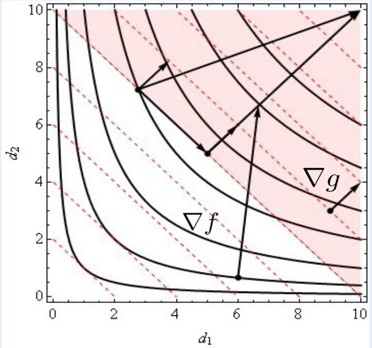
- ✓ Can use existing simulation tools
- ✓ Non-intrusive for code
- ✓ Surrogate has gradients
- • ML training can be expensive
- • Hard to modify design space
- • Limited to O(10) design parameters




**Present (Data Driven)**

## Gradient-Based

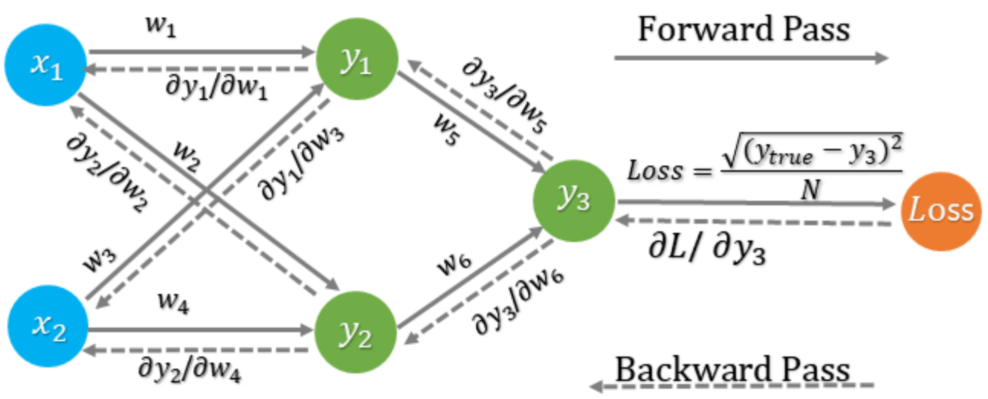
- ✓ Large parameter space, O(1M)
- ✓ Fast convergence
- ✓ Agile parameterizations
- ✓ Provable local optimality
- • Requires gradients
- • Code Intrusive




**Future**

# Backpropagation and automatic differentiation are powerful techniques we can adapt from the ML community

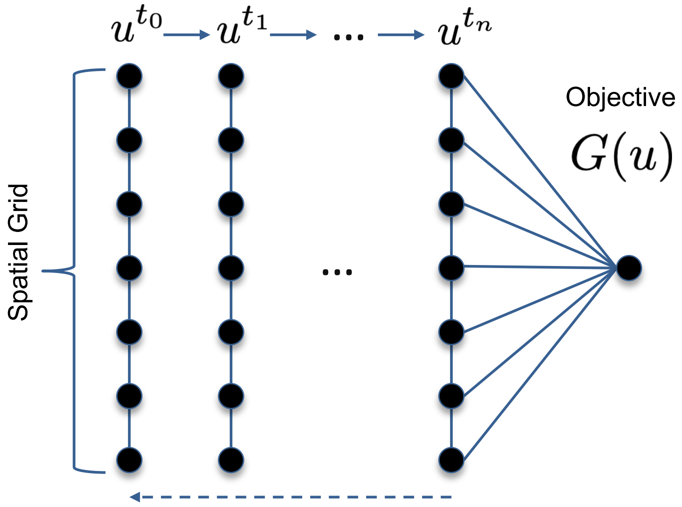
Neural Network Trained using Back Propagation




From: Physics-Based Deep Learning for Flow Problems http://dx.doi.org/10.3390/en14227760

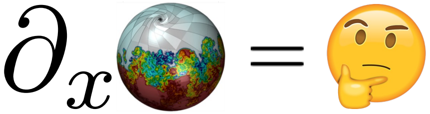
- AI/ML revolution powered by backpropagation/automatic differentiation (AD) on complex models
- We can leverage these techniques for generalized adjoint solvers for non-linear multiphysics

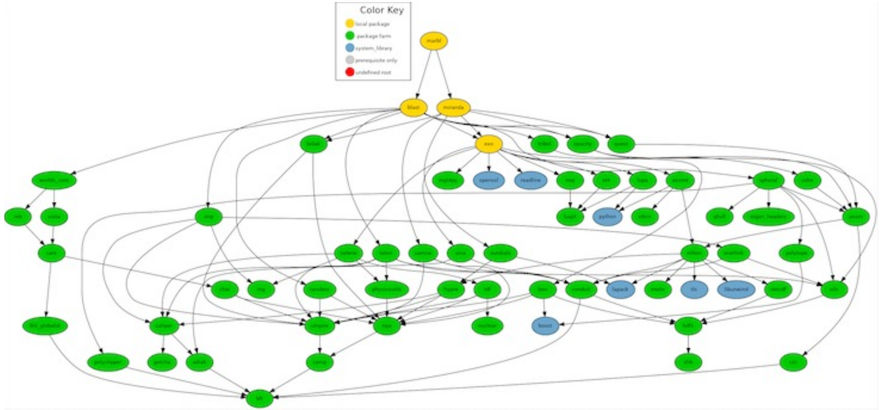
Simulation Gradients using Back Propagation




$$\frac{\partial G}{\partial u^{t_0}} = \frac{\partial G}{\partial u^{t_n}} \frac{\partial u^{t_n}}{\partial u^{t_{n-1}}} \cdots \frac{\partial u^{t_2}}{\partial u^{t_1}} \frac{\partial u^{t_1}}{\partial u^{t_0}}$$

# Enabling AD in a multiphysics code stack will be a grand challenge

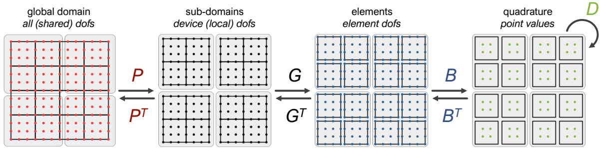





- Multiphysics codes are complex software stacks
- We need to decompose the problem into manageable pieces

- Partially assembled FE methods offer a natural insertion point for AD

$$\boxed{A = P^T G^T B^T D B G P}$$




- Need to differentiate at Q-points only!

$$\nabla_u A(u;\rho) = P^T G^T B^T \nabla_{\hat{u}} D(\hat{u},\hat{\rho})$$

- Differentiate the Q-function *D* with AD tools

LLNL mfem project is exploring AD using the Enzyme library

# Enabling AD in a multiphysics code stack will be a grand challenge … similar to our GPU transition

- Refactoring an existing, long-standing code base is a daunting challenge
  - GPUs were a disruptive technology for our existing codes
- We have successfully refactored multiple, large scale production codes for GPUs
- Integrating AD into our multiphysics codes will be a similar challenge
  - We have new codes with AD built in from the start (LLNL Smith project)
  - We will need to integrate AD into all 3rd party libraries
  - We will need software abstraction layers, much like performance portability abstractions
    - 3rd party library APIs updated to include passing gradients
  - One size will likely not fit all, we need to embrace multiple paths
    - AD at compiler level, dual numbers, finite differences, analytic gradients

Smith

**We can meet the challenge using the same approaches which brought success on GPUs**

# Conclusions and future outlook

- We have invested in single source, performance portable HPC multiphysics codes over the past 7 years
- We have achieved breakthrough performance gains on GPU architectures
  - We intend to capitalize on these gains in the post exascale era
- There are appealing aspects of cloud compute in combination with on-premise HPC
- Our pursuit of computational performance is not stopping
- We believe the key to unlocking future performance gains will come from gradient based optimization