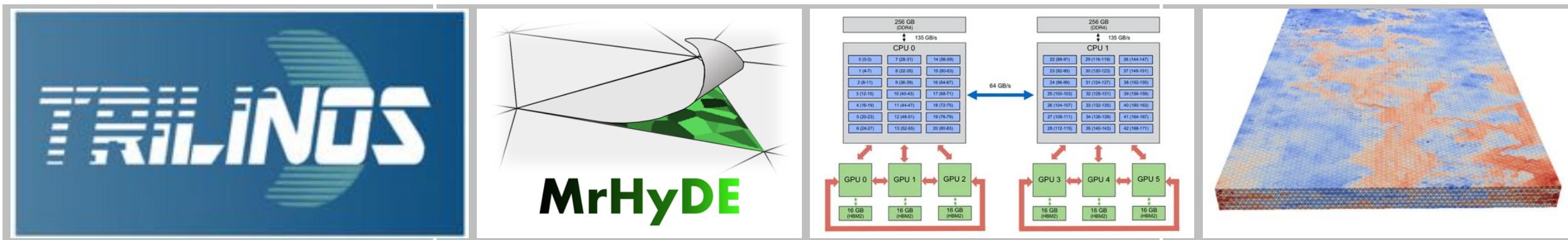


# Introduction to Trilinos and MrHyDE

MrHyDE = {M}ulti-{r}esolution {Hy}bridized {D}ifferential {E}quations



Tim Wildey

Optimization and Uncertainty Quantification Department  
Center for Computing Research

**Any questions from yesterday?**

# Tutorial Outline

## Day 1 - Introduction to Trilinos

- High-level overview of Trilinos
  - *An appropriate build of Trilinos will be available for anyone on the HPC systems. We will not be building Trilinos in this session. If someone does not have access to the HPC systems, I will work with them beforehand to get a build of Trilinos on their Mac or Linux machine.*
- Deeper dive into Kokkos and Sacado.
  - *A basic understanding of these packages will be helpful for day 2.*
- Exercise: creating and working with arrays (Kokkos Views) and automatic differentiation objects (Sacado AD)

## Day 2 - Introduction to MrHyDE

- High-level overview of MrHyDE
- How to download, compile, run and visualize results
- Exercise: adding a new PDE in MrHyDE

## Day 3 - More advanced features in Trilinos/MrHyDE

- Hybridized methods and concurrent multiscale modeling
- Solving coupled multiphysics problems
- Performance portability and using heterogeneous computational architectures
- Large-scale PDE constrained optimization

# Tutorial Outline

## Day 1 - Introduction to Trilinos

- High-level overview of Trilinos
  - *An appropriate build of Trilinos will be available for anyone on the HPC systems. We will not be building Trilinos in this session. If someone does not have access to the HPC systems, I will work with them beforehand to get a build of Trilinos on their Mac or Linux machine.*
- Deeper dive into Kokkos and Sacado.
  - *A basic understanding of these packages will be helpful for day 2.*
- Exercise: creating and working with arrays (Kokkos Views) and automatic differentiation objects (Sacado AD)

## Day 2 - Introduction to MrHyDE

- High-level overview of MrHyDE
- How to download, compile, run and visualize results
- Exercise: adding a new PDE in MrHyDE

## Day 3 - More advanced features in Trilinos/MrHyDE

- Hybridized methods and concurrent multiscale modeling
- Solving coupled multiphysics problems
- Performance portability and using heterogeneous computational architectures
- Large-scale PDE constrained optimization

# **Hybridized Methods and Concurrent Multiscale Modeling**

# Mixed Formulation of Poisson

Suppose we want to solve Poisson equation on  $\Omega \subset \mathbb{R}^d$  using a mixed method.

$$\begin{aligned}\mathbf{u} &= -\mathbf{K}\nabla p, \\ \nabla \cdot \mathbf{u} &= f\end{aligned}$$

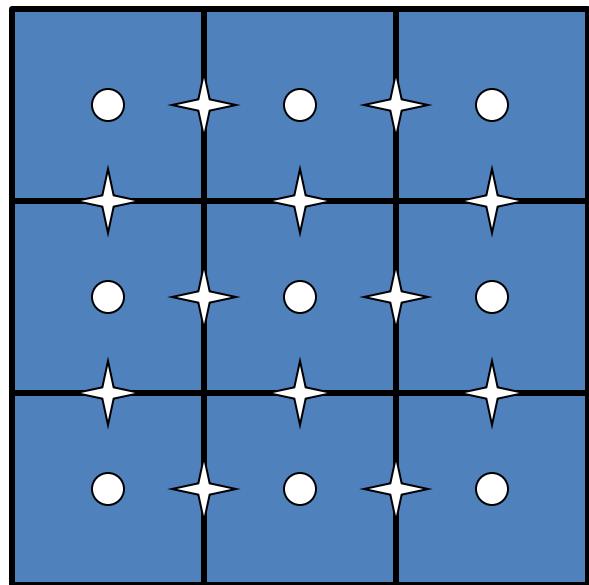
Variational formulation seek  $(\mathbf{u}, p) \in H(\text{div}, \Omega) \times L^2(\Omega)$  such that,

$$\begin{aligned}(\mathbf{K}^{-1}\mathbf{u}, \mathbf{v}) - (p, \nabla \cdot \mathbf{v}) &= \mathbf{0}, \\ (\nabla \cdot \mathbf{u}, q) &= (f, q)\end{aligned}$$

for all  $(\mathbf{v}, q) \in H(\text{div}, \Omega) \times L^2(\Omega)$ .

Discrete approximation spaces must be chosen carefully

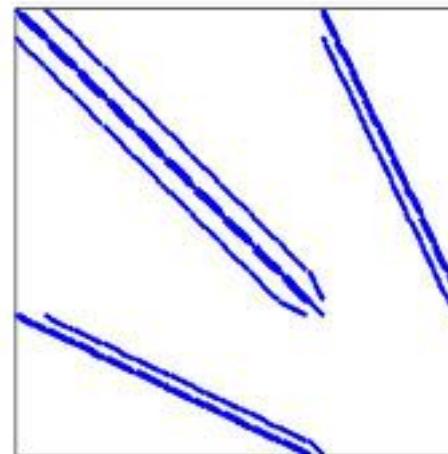
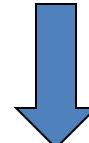
# Hybridization of Mixed Methods



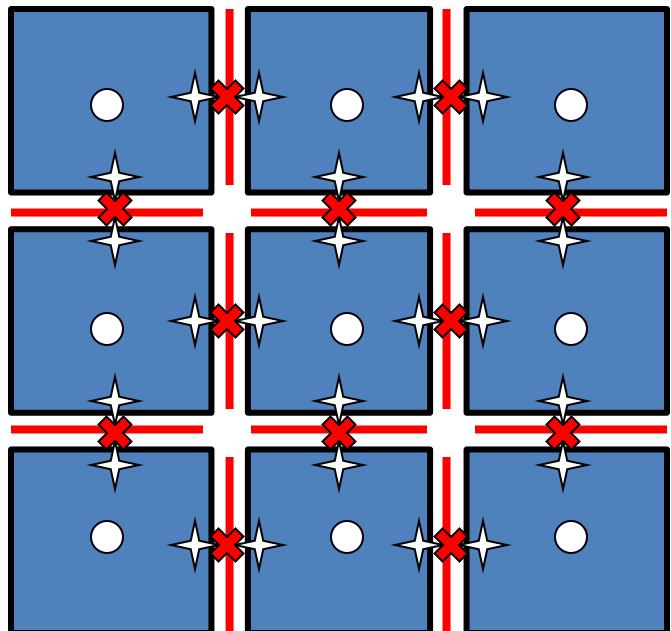
○ | Pressure degree of freedom  
★ | Velocity degree of freedom

Mixed methods yield linear systems of the form:

$$\begin{pmatrix} M & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ -f \end{pmatrix}$$



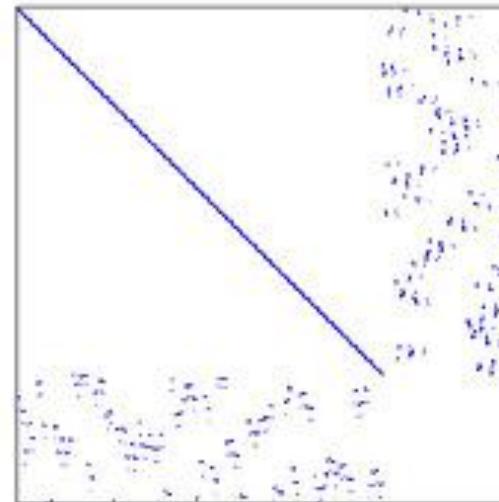
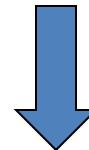
# Hybridization of Mixed Methods



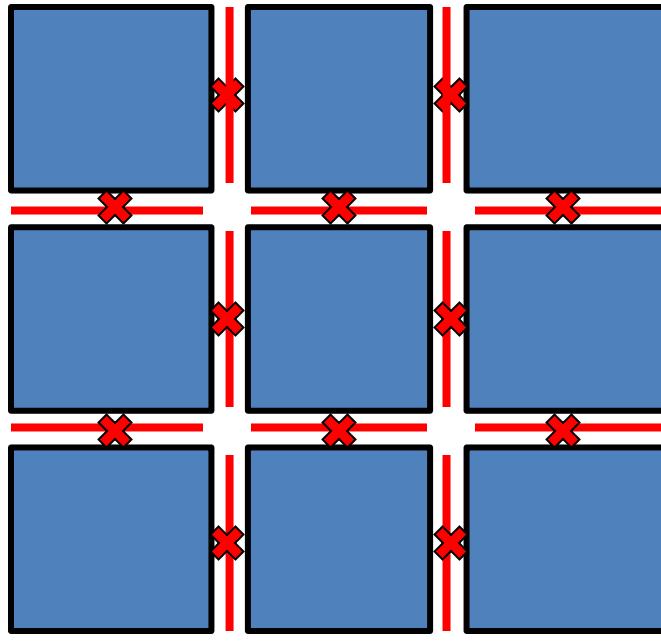
- | Pressure degree of freedom
- ★ | Velocity degree of freedom
- ✖ | Lagrange multiplier dof

Introduce Lagrange multipliers on the element boundaries:

$$\begin{pmatrix} M & B & C \\ B^T & 0 & 0 \\ C^T & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ -f \\ 0 \end{pmatrix}$$



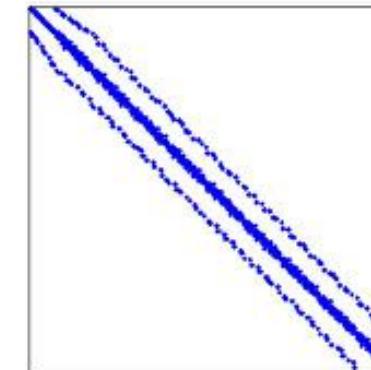
# Hybridization of Mixed Methods



✖ | Lagrange multiplier dof

Reduce to Schur complement for  
Lagrange multipliers:

$$A\lambda = g$$



# Hybridized Mixed Formulation of Poisson

Let  $\mathcal{T}_h$  denote the mesh.

Let  $\mathcal{E}_h$  denote the “skeleton” of the mesh, i.e., the interior edges/faces.

The modified variational formulations seeks  $\lambda \in H^{1/2}(\mathcal{E}_h)$  such that

$$\sum_{T_i \in \mathcal{T}_h} -\langle \mathbf{u}_i(\lambda) \cdot \mathbf{n}_i, \mu \rangle_{\partial T_i} = 0, \quad \forall \mu \in H^{1/2}(\mathcal{E}_h).$$

where  $(\mathbf{u}_i, p_i) \in H(\text{div}, T_i) \times L^2(T_i)$  satisfy

$$\begin{aligned} (\mathbf{K}^{-1} \mathbf{u}_i, \mathbf{v}_i)_{T_i} - (p_i, \nabla \cdot \mathbf{v}_i)_{T_i} &= -\langle \lambda, \mathbf{v}_i \cdot \mathbf{n}_i \rangle_{\partial T_i}, \\ (\nabla \cdot \mathbf{u}_i, q_i)_{T_i} &= (f, q_i)_{T_i} \end{aligned}$$

for all  $(\mathbf{v}_i, q_i) \in H(\text{div}, T_i) \times L^2(T_i)$ .

The mapping  $\lambda \rightarrow \mathbf{u}_i \cdot \mathbf{n}_i$  is often called the Dirichlet-to-Neumann (DtN) map

# Hybridized Methods Are Appealing for High-order Discretizations

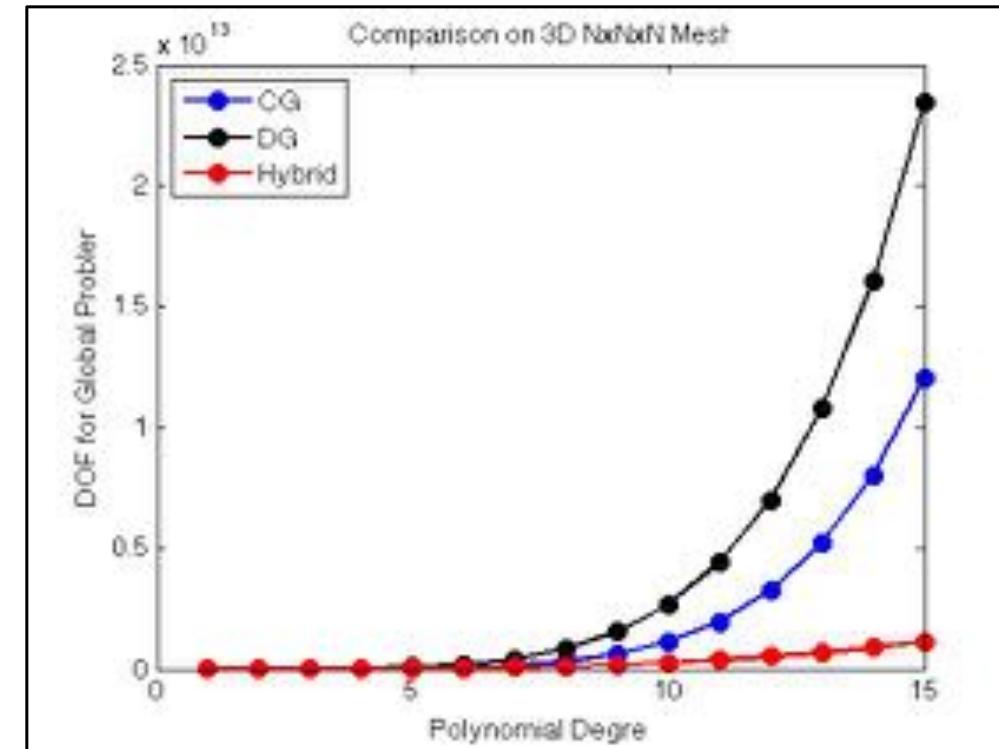
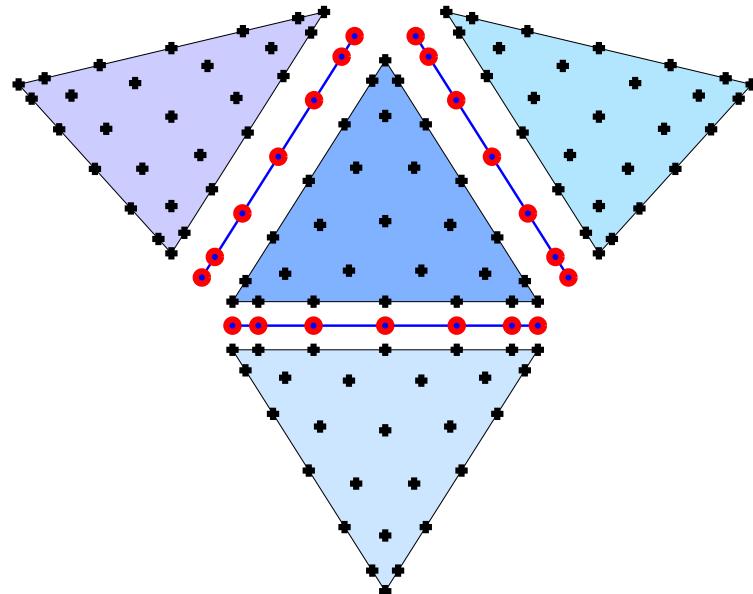
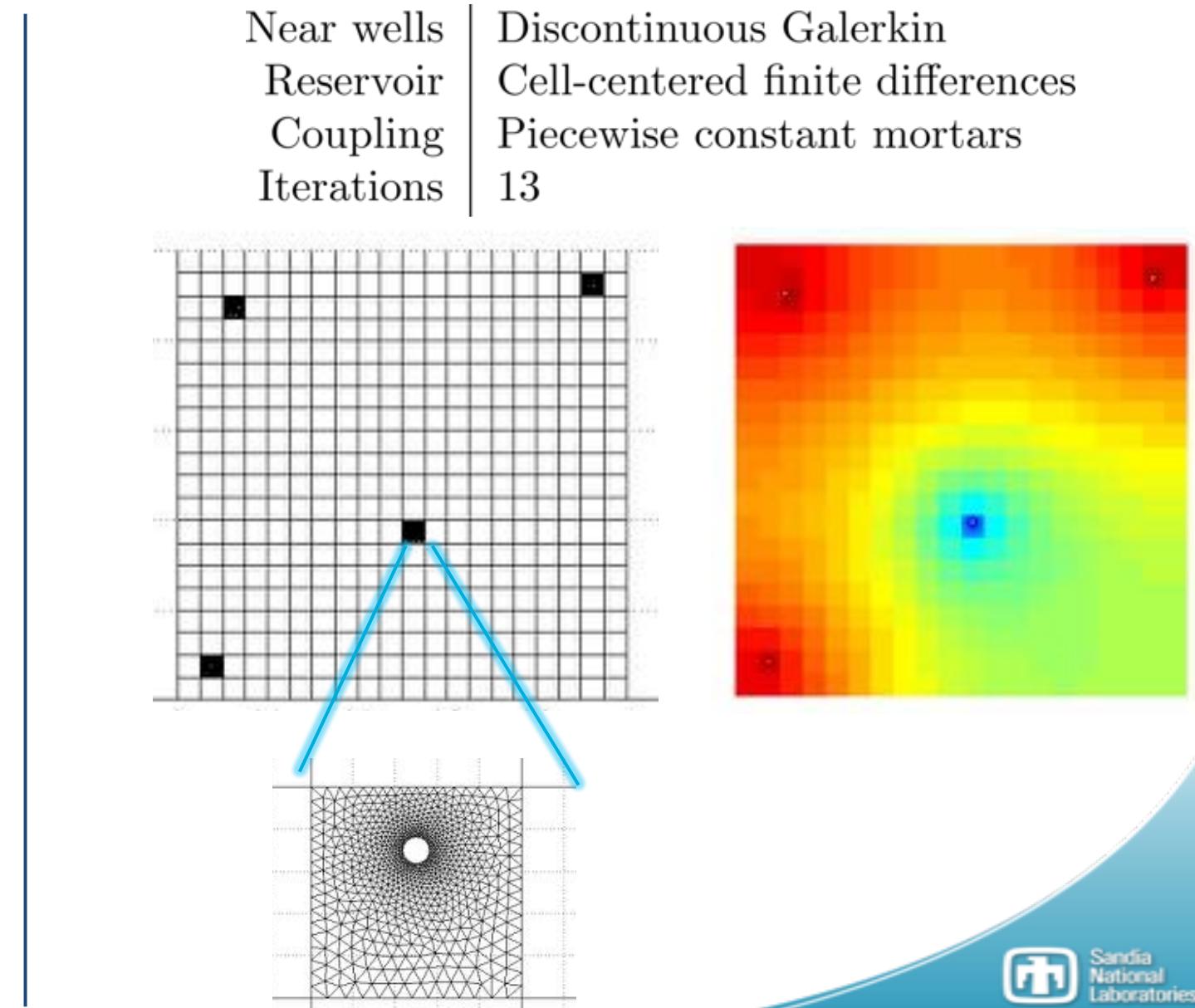


Figure: Hybridization schematic for a high-order DG method (left) and a comparison of the number of coupled DOFs for CG, DG, and HDG (right).

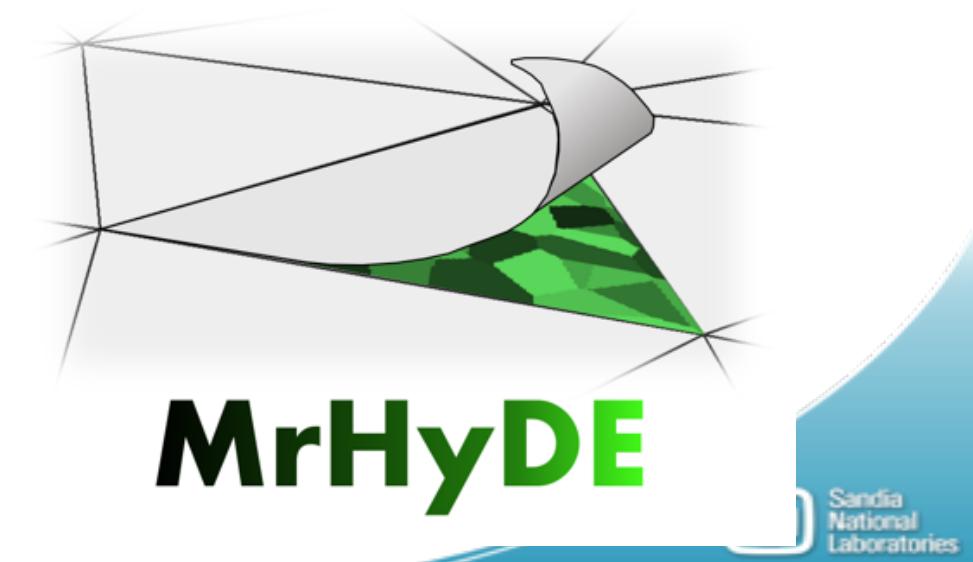
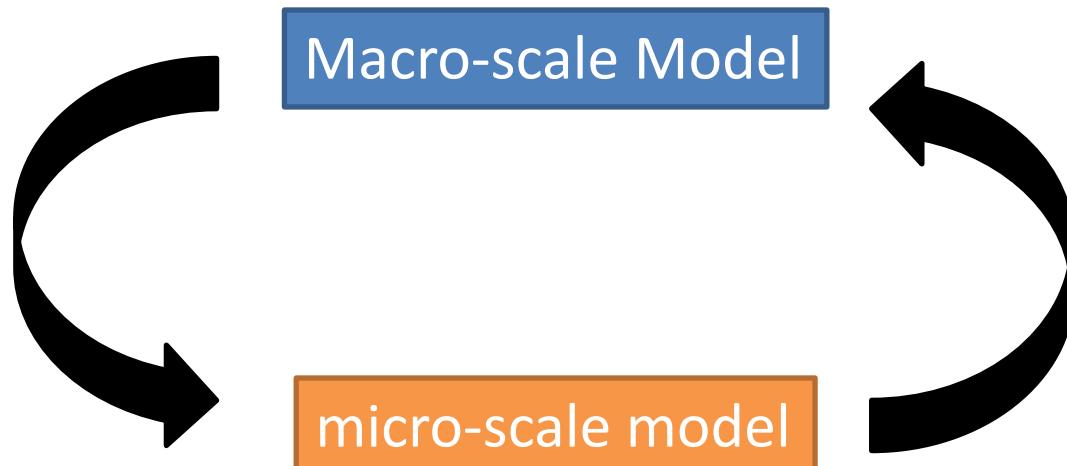
# Hybridized Methods Enable Multi-Scale, Multi-Physics and Multi-Numerics

Model	Laplace equation
Domain	$(0, 1) \times (0, 1)$
Method	DG-NIPG <sub>1</sub> (Yellow)
Method	Mixed-RT <sub>1</sub> (Pink)
Mesh	Triangles
True solution	$p = xye^{x^2y^3}$
Tolerance	$1 \times 10^{-6}$



# Why Aren't They Used More Often?

- Implementation is certainly nontrivial
  - Global interface problem using basis functions on the skeleton of the mesh
  - Local subgrid problems require additional variables and DOF managers
  - Visualizing the solutions can be difficult
  - Implicit methods require not only the DtN map, but also the Jacobian w.r.t interface variables (finite difference approximations are usually used and these are expensive/inaccurate)
  - More DOFs for low-order approximations (unless using a continuous basis as in embedded DG)
- MrHyDE implements these methods through a concurrent multiscale framework



# An Example from the Regression Suite



```
version: 1.0.0
grid_type: subgrid
grid_size: 100
grid_resolution: 1.0
grid_center: [50, 50]
grid_min: [0, 0]
grid_max: [100, 100]
grid_type: subgrid
grid_size: 100
grid_resolution: 1.0
grid_center: [50, 50]
grid_min: [0, 0]
grid_max: [100, 100]
```

input.yaml

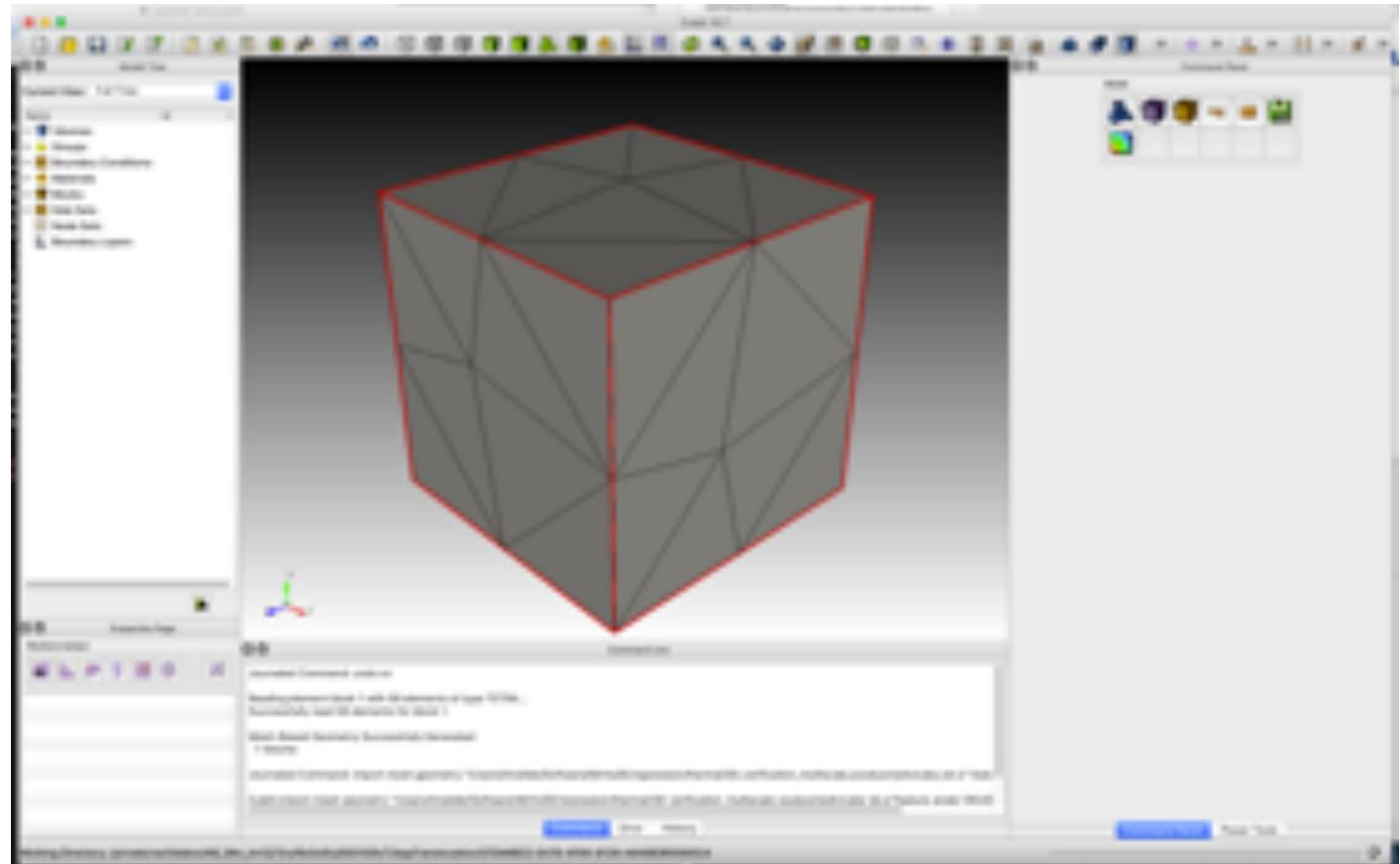


```
version: 1.0.0
grid_type: subgrid
grid_size: 100
grid_resolution: 1.0
grid_center: [50, 50]
grid_min: [0, 0]
grid_max: [100, 100]
grid_type: subgrid
grid_size: 100
grid_resolution: 1.0
grid_center: [50, 50]
grid_min: [0, 0]
grid_max: [100, 100]
```

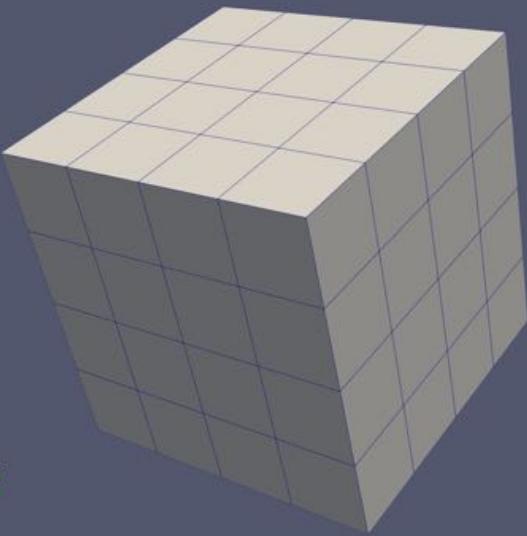
subgrid\_input.yaml

# A Unstructured Multimodel Example

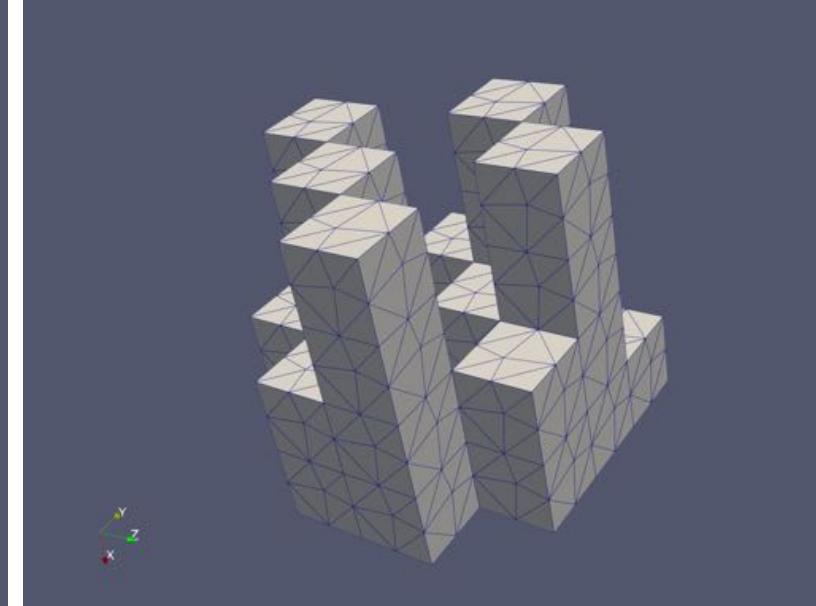
- Multiple options for defining a subgrid mesh
  - Inline
  - Panzer
  - Exodus
- These are defined on the reference element for the macro-scale mesh
- This gets mapped to the physical elements for the subgrid meshes
- Different elements can have different subgrid models
  - They don't talk directly to each other



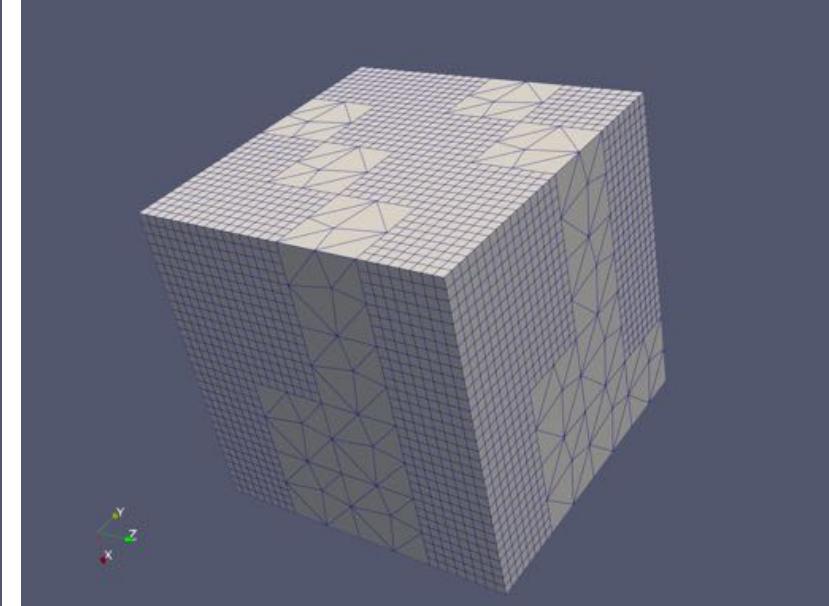
# A Unstructured Multimodel Example



Coarse mesh

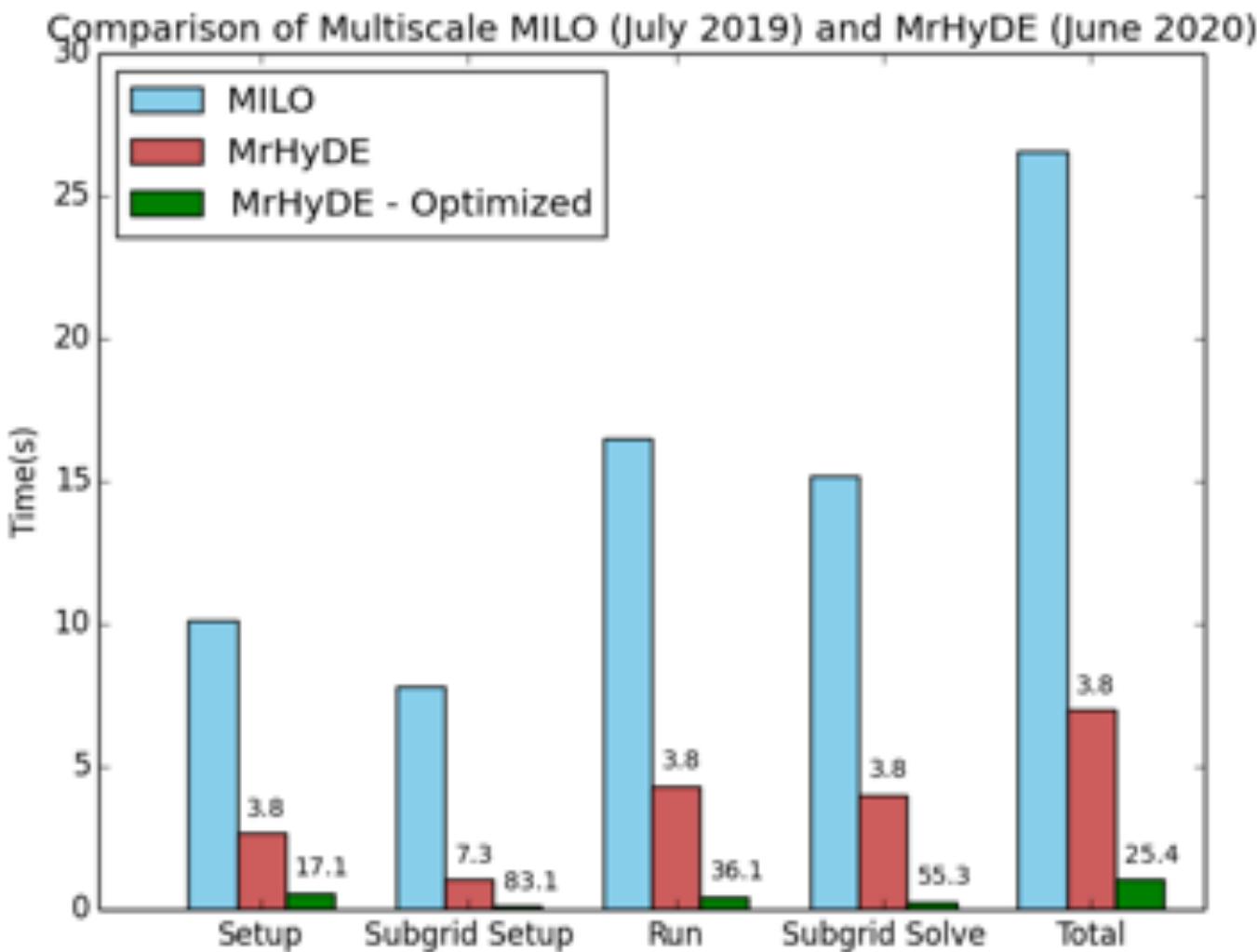


Some subgrids use the exodus mesh



Some subgrids use a panzer mesh

# Performance of Multiscale Framework



# Solving Coupled Multiphysics Problems in MrHyDE

# Solving Coupled Multiphysics Problems in MrHyDE

Navier Stokes:

$$\frac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = \mathbf{f}$$
$$\nabla \cdot \mathbf{u} = 0$$

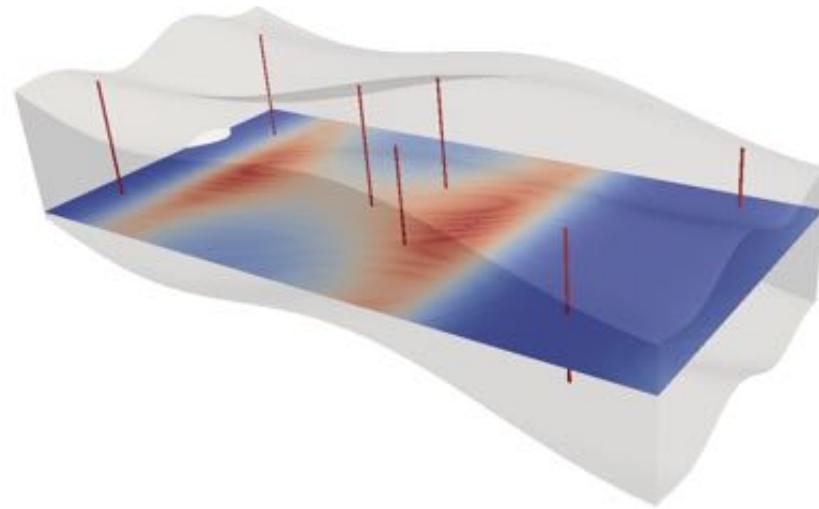
Convection-Diffusion-Reaction:

$$\frac{\partial c}{\partial t} - \epsilon \Delta c + \mathbf{v} \cdot \nabla c + R(c) = s(x, t)$$

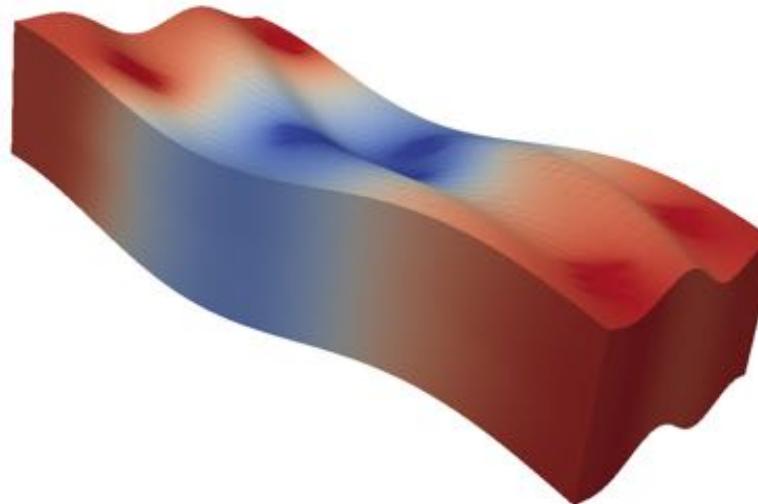
Setting  $\mathbf{v} = \mathbf{u} \rightarrow \{$

```
Physics:  
modules: 'navier stokes, cdr'  
Dirichlet conditions:  
ux:  
    bottom: '0.0'  
    top: '0.0'  
uy:  
    bottom: '0.0'  
    top: '0.0'  
c:  
    bottom: '0.0'  
    top: '0.0'  
Discretization:  
order:  
    ux: 1  
    uy: 1  
    pr: 1  
    c: 1  
quadrature: 2  
Functions:  
source ux: '1.0'  
source: 'exp(bubble)'  
diffusion: '0.01'  
xvel: 'ux'  
yvel: 'uy'  
reaction: '0.0'  
SUPG tau: '0.0'  
bubble: '-10.0*(x-2)*(x-2) - 10.0*(y-0.5)*(y-0.5)'
```

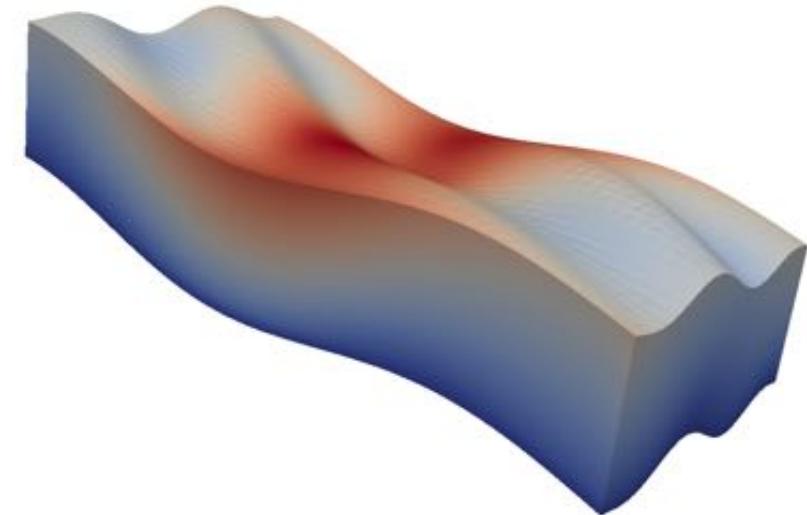
# Solving Coupled Multiscale/Multiphysics Problems in MrHyDE



Computational domain with wells and slice of permeability field generated from a realization of a KL expansion



Pressure field from single-phase slightly compressible model using multiscale discretization.



Magnitude of the displacement using a single-scale Biot poroelastic model.

# **Performance Portability and Heterogeneous Architectures**

# Solving Coupled Multiphysics Problems in MrHyDE

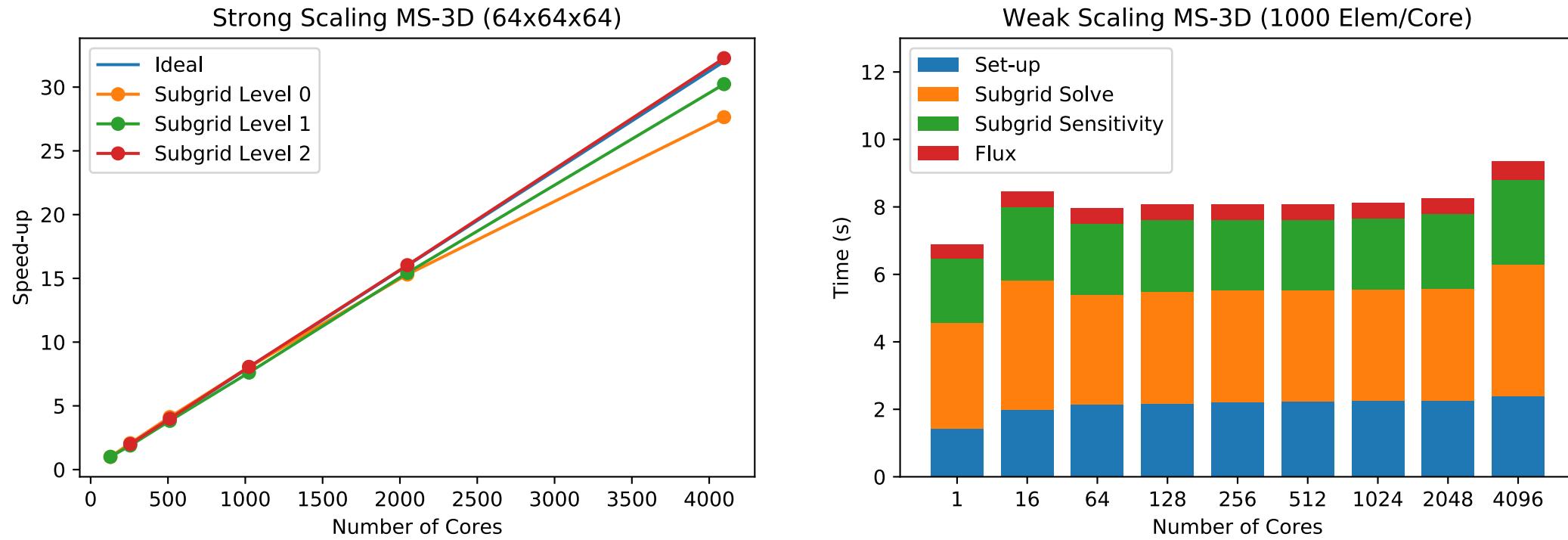
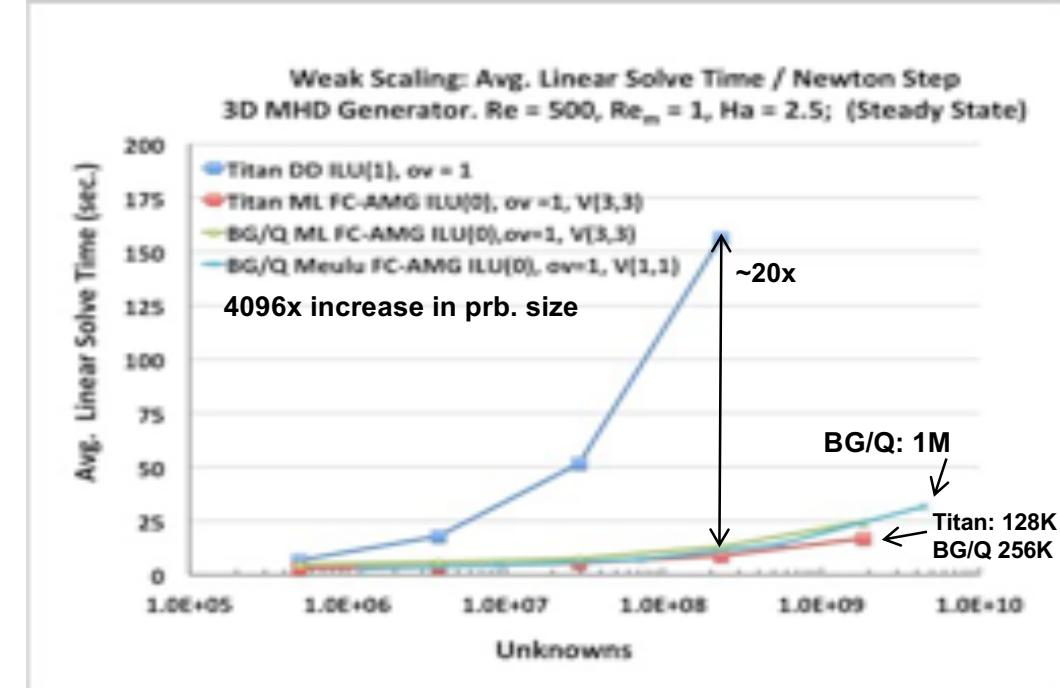
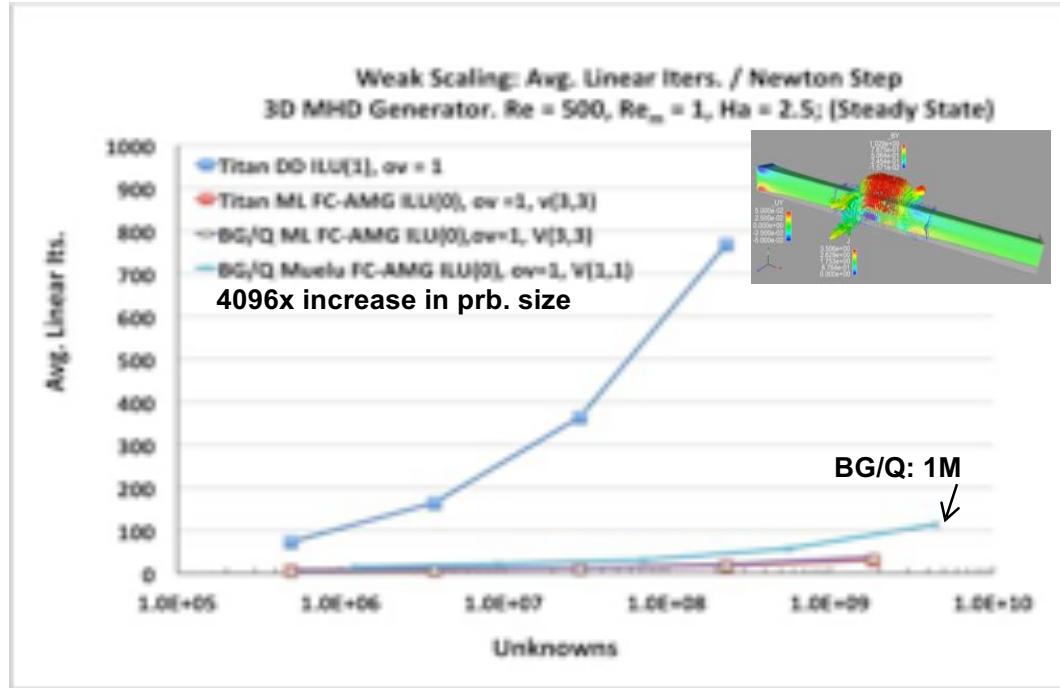


Figure: Strong and weak scaling for subgrid models in MrHyDE.

# Trilinos Solvers Scale to Millions of Cores

Courtesy of J. Shadid. Computational results from Drekar



**Figure:** Weak scaling of Trilinos solvers in Drekar out to 13B DOF (MHD) and 40B DOF MHD (steady) weak scaling studies to 128K Cray XK7, 1M BG/Q (CFD) on 128K cores on Cray XK7 and 4.1B DOF on 1.6M cores on BG/Q.

# Modern Computational Architectures are Heterogeneous

TOP500 Sites for November 2019

For more information about the sites and systems in the list, click on the links or view the comparison.



Rank	Name	Type	System	Score	Rank	Power
1	Summit	Supercomputer	IBM Power System AC922, IBM Power9, 2x256GB DDR4, 6xV100, 4.608 nodes	1000.000	1	10.000
2	Sierra	Supercomputer	IBM Power System AC922, IBM Power9, 2x256GB DDR4, 6xV100, 4.608 nodes	1000.000	2	10.000
3	Kazan	Supercomputer	Siemens S9000, Siemens Xeon E5-2695 v4, 1.000 nodes	1000.000	3	10.000
4	Frontier	Supercomputer	Cray XC40, Intel Xeon Gold 6138, 1.000 nodes	1000.000	4	10.000
5	Aztec	General Purpose	Dell PowerEdge R740xd, Intel Xeon E5-2695 v4, 1.000 nodes	1000.000	5	10.000

Graphic and more detailed schematic of Summit node can be found at:  
<https://gitlab.com/petsc/petsc/-/wikis/PETSc-on-GPUs>

## Summit Specs:

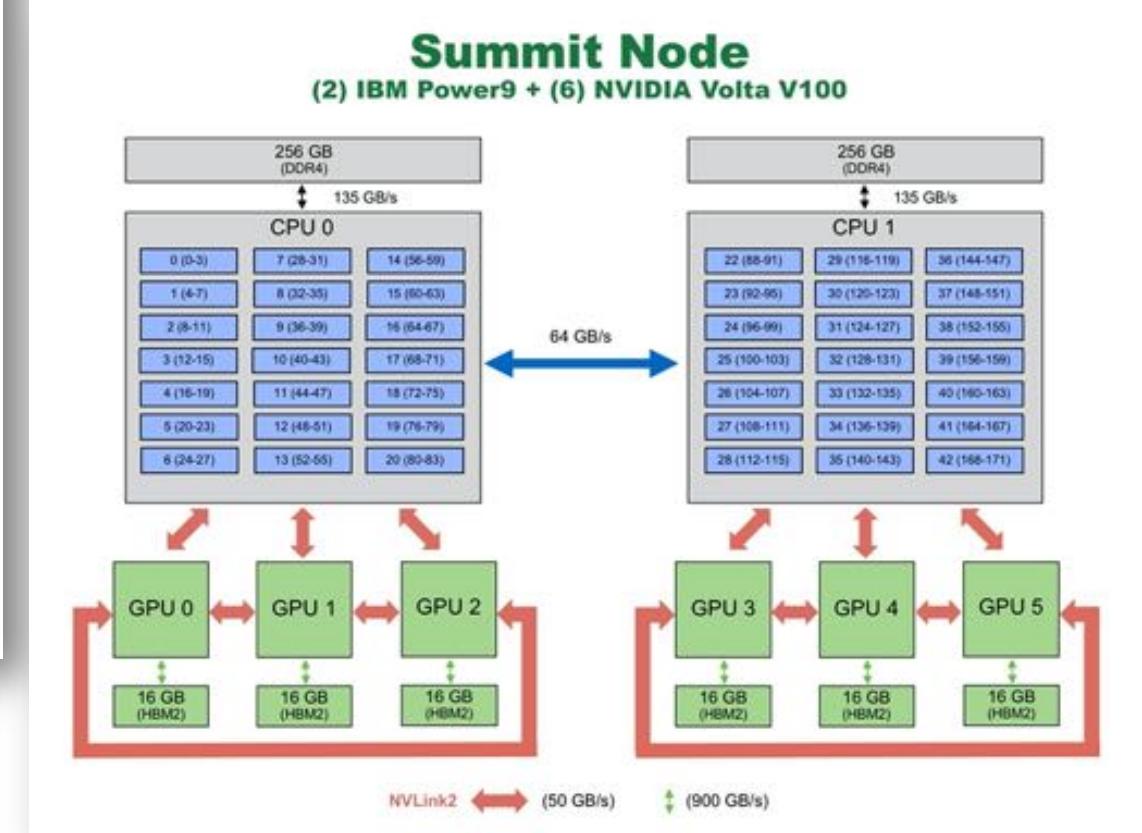
**Processor:** IBM POWER9™ (2/node)

**GPUs:** 27,648 NVIDIA Volta V100s (6/node)

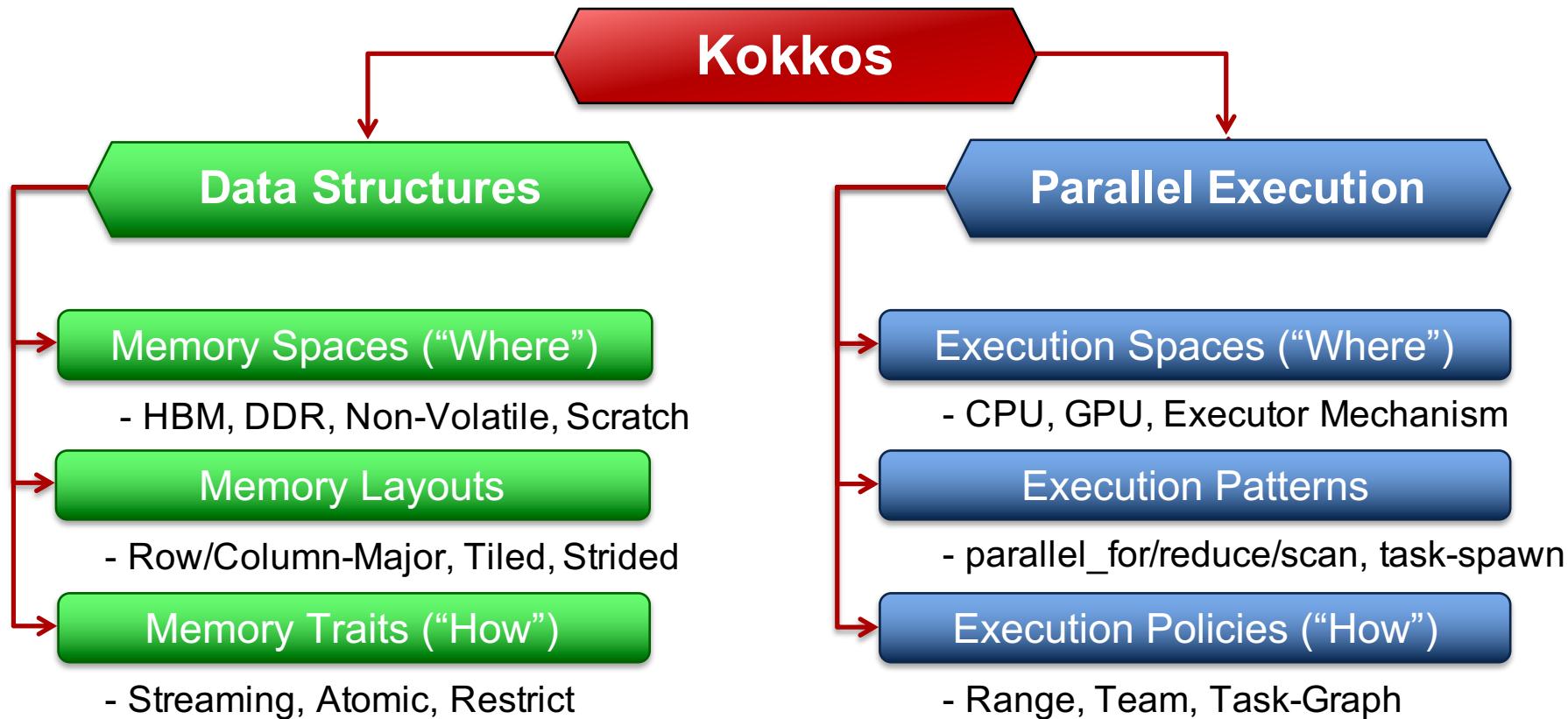
**Nodes:** 4,608

**Node Performance:** 42TF

**Memory/node:** 512GB DDR4 + 96GB HBM2



# Kokkos Library for Performance Portability



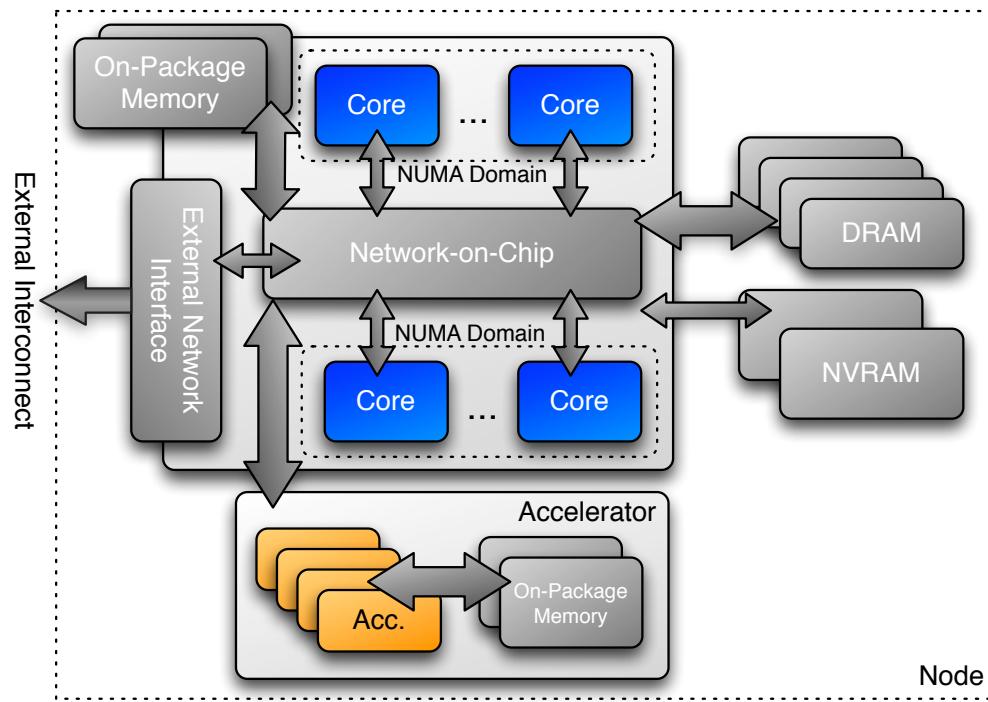
Kokkos is not the only library designed on this model. RAJA from LLNL is very similar.  
See [https://raja.readthedocs.io/en/master/getting\\_started.html](https://raja.readthedocs.io/en/master/getting_started.html)

# What is an Execution Space?

Execution spaces (1)

## Execution Space

a homogeneous set of cores and an execution mechanism  
(i.e., “place to run code”)



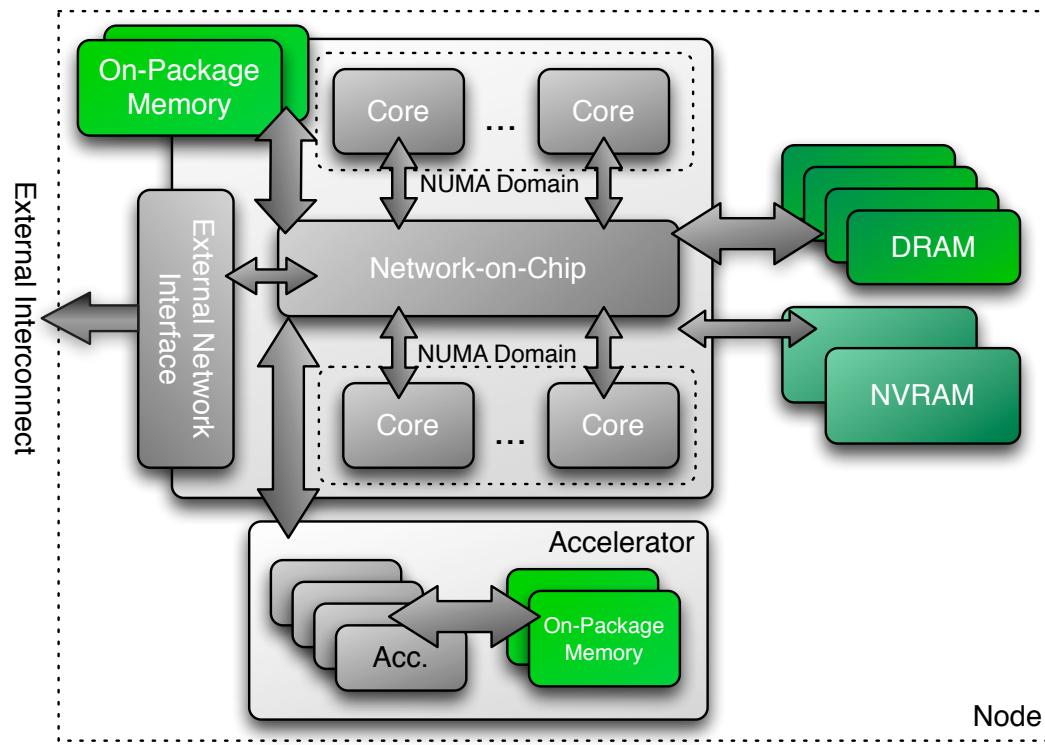
Execution spaces: Serial, Threads, OpenMP, Cuda, ROCm, ...

# What is a Memory Space?

Memory spaces (0)

## Memory space:

explicitly-manageable memory resource  
(i.e., “place to put data” )



# Creating Views in a Specific Memory Space

Memory spaces (1)

## Important concept: Memory spaces

Every view stores its data in a **memory space** set at compile time.

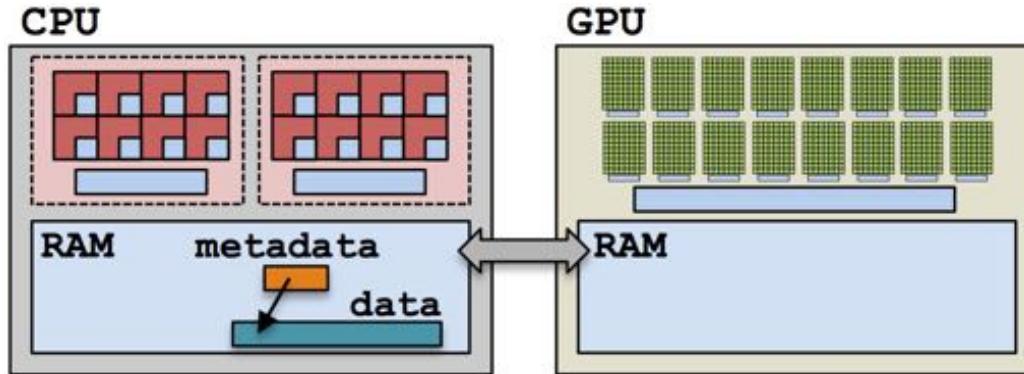
- ▶ `View<double***, MemorySpace> data(...);`
- ▶ Available **memory spaces**:  
    `HostSpace`, `CudaSpace`, `CudaUVMSpace`, ... more
- ▶ Each **execution space** has a default memory space, which is used if **Space** provided is actually an execution space
- ▶ If no Space is provided, the view's data resides in the **default memory space** of the **default execution space**.

# Creating Views in a Specific Memory Space

Memory spaces (2)

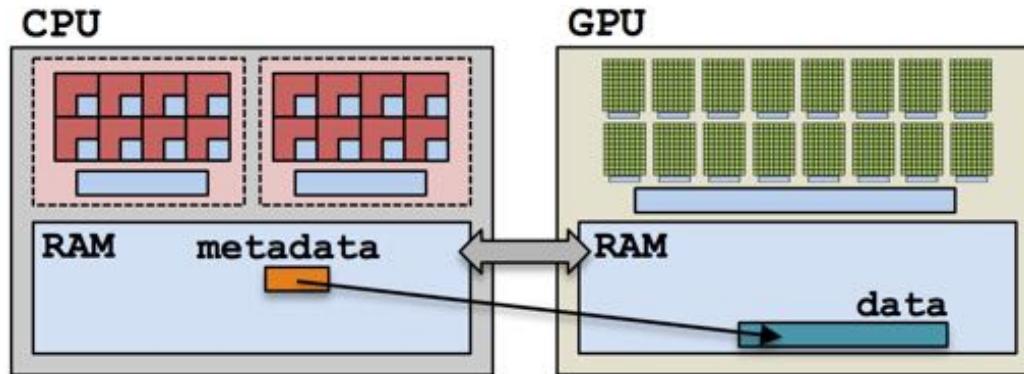
## Example: HostSpace

```
View<double**, HostSpace> hostView(...constructor arguments...);
```



## Example: CudaSpace

```
View<double**, CudaSpace> view(...constructor arguments...);
```



# Data Layouts Do Matter

Layout

## Important concept: Layout

Every View has a multidimensional array Layout set at compile-time.

```
View<double***, Layout, Space> name(...);
```

- ▶ Most-common layouts are LayoutLeft and LayoutRight.
  - LayoutLeft: left-most index is stride 1.
  - LayoutRight: right-most index is stride 1.
- ▶ If no layout specified, default for that memory space is used.
  - LayoutLeft for CudaSpace, LayoutRight for HostSpace.
- ▶ Layouts are extensible: ~50 lines
- ▶ Advanced layouts: LayoutStride, LayoutTiled, ...

# Enabling Threaded Data Parallelism

## Pattern

```
for (element = 0; element < numElements; ++element) {  
    total = 0;  
    for (qp = 0; qp < numQPs; ++qp) {  
        total += dot(left[element][qp], right[element][qp]);  
    }  
    elementValues[element] = total;  
}
```

## Body

## Policy

### Terminology:

- ▶ **Pattern**: structure of the computations  
for, reduction, scan, task-graph, ...
  - ▶ **Execution Policy**: how computations are executed  
static scheduling, dynamic scheduling, thread teams, ...
  - ▶ **Computational Body**: code which performs each unit of work; e.g., the loop body
- ⇒ The **pattern** and **policy** drive the computational **body**.

# Enabling Threaded Data Parallelism

Kokkos parallel execution:

```
Custom parallel_for("Label",
  RangePolicy< ExecutionSpace >(0,numberOfIntervals),
  [=] (const int64_t i) {
    /* ... body ... */
});
```

MrHyDE uses Kokkos parallel executions extensively.

For example, the volume residual from shallow water equations:

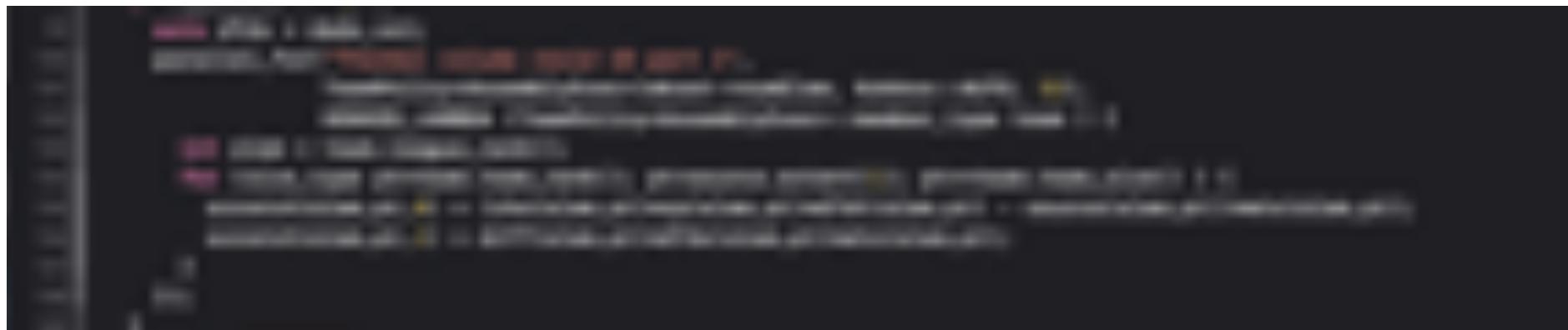


# Enabling Hierarchical Parallelism

Sometimes we need even more parallelism:

```
parallel_something(
    TeamPolicy<ExecutionSpace>(numberOfTeams, Kokkos::AUTO),
    /* functor */);
```

MrHyDE uses hierarchical parallelism especially when working with Views of AD objects.  
For example, the a piece of volume residual from the thermal equation:



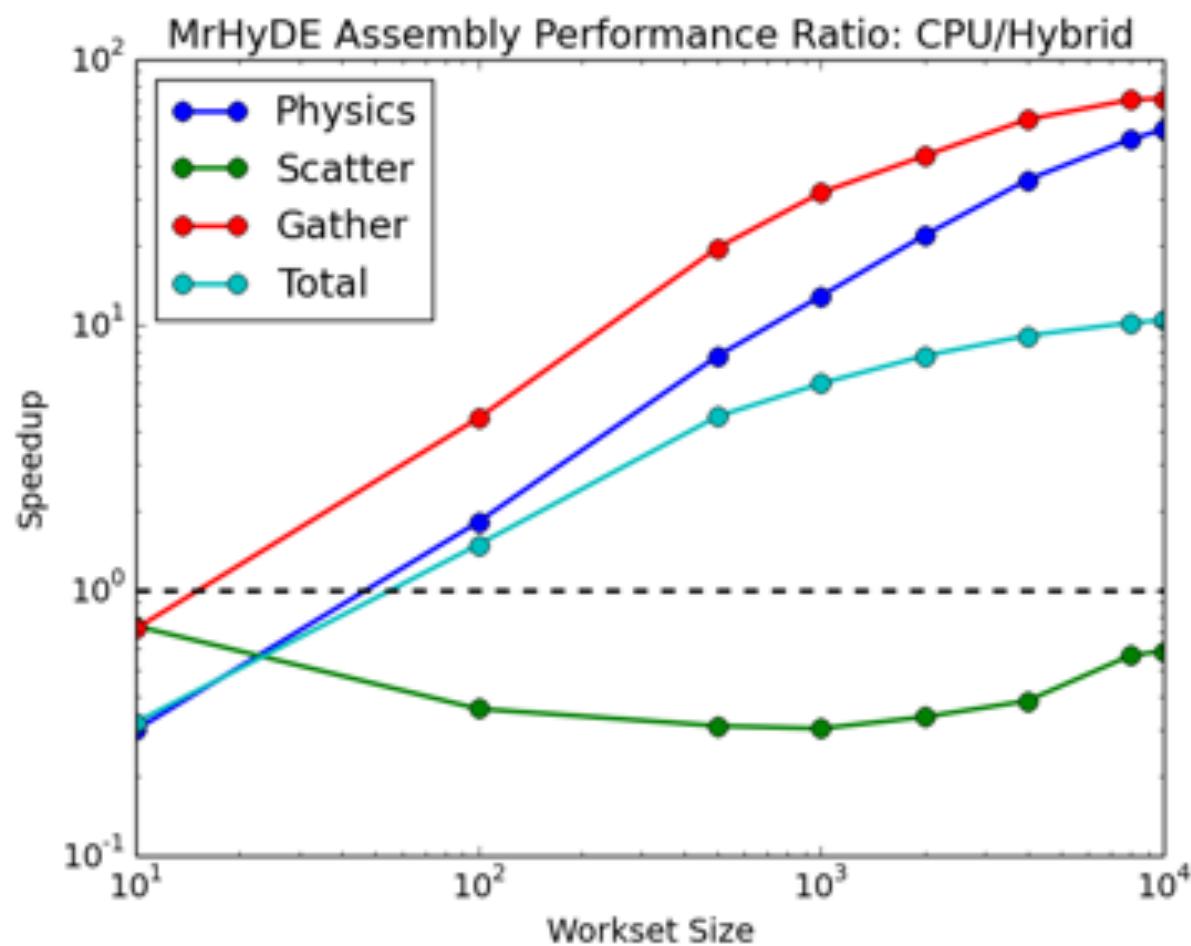
This is actually 3 levels of parallelism (thread, team, vector)

# MrHyDE Provides a Few Options

Host	Assembly	Solver	Subgrid Assembly	Subgrid Solver
CPU	CPU	CPU	CPU	CPU
CPU	GPU	CPU	GPU	GPU
CPU	GPU	GPU	GPU	GPU
CPU	CPU	GPU	CPU	CPU

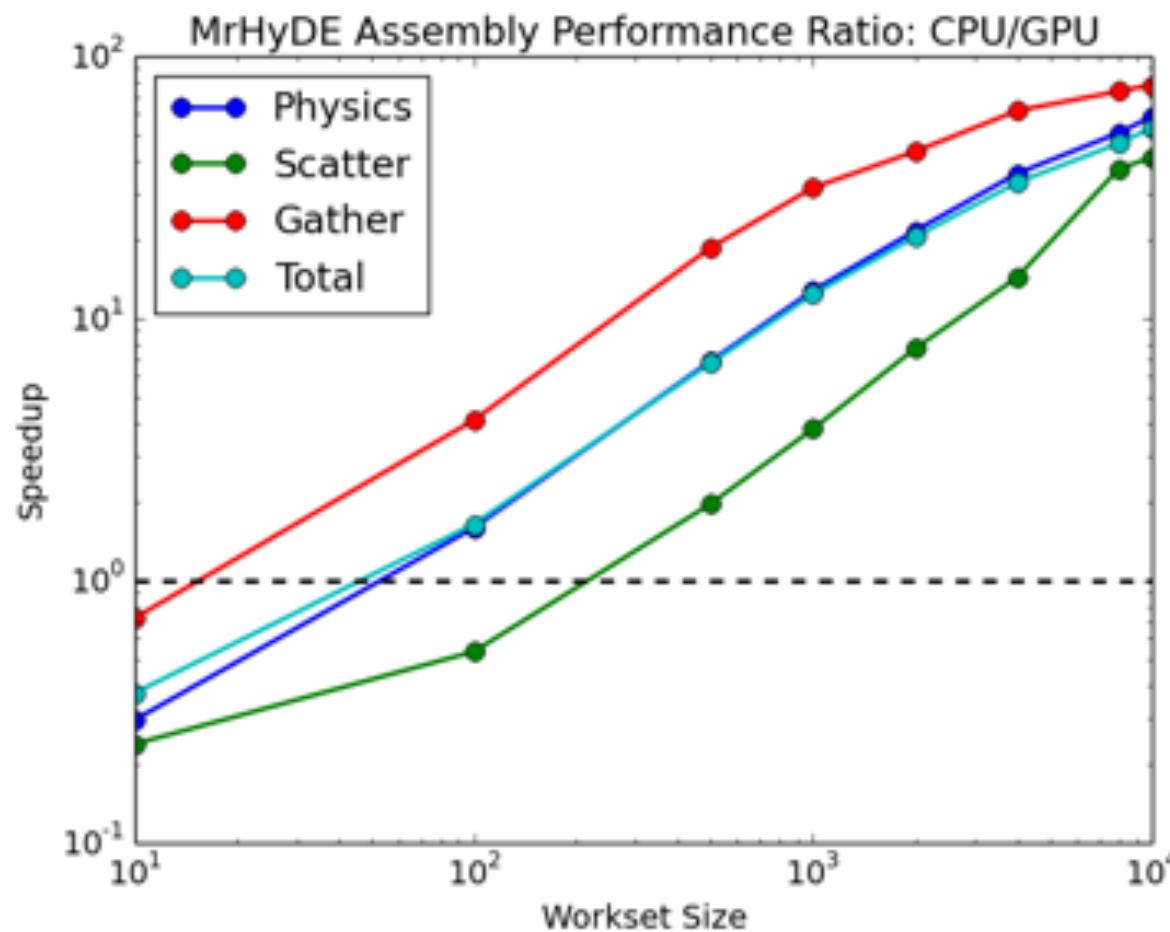
Available options when running on a heterogeneous architecture, e.g., white, weaver, sierra, summit.  
Last row has not been tested and is probably a bad idea.

# MrHyDE Assembly: CPU vs GPU



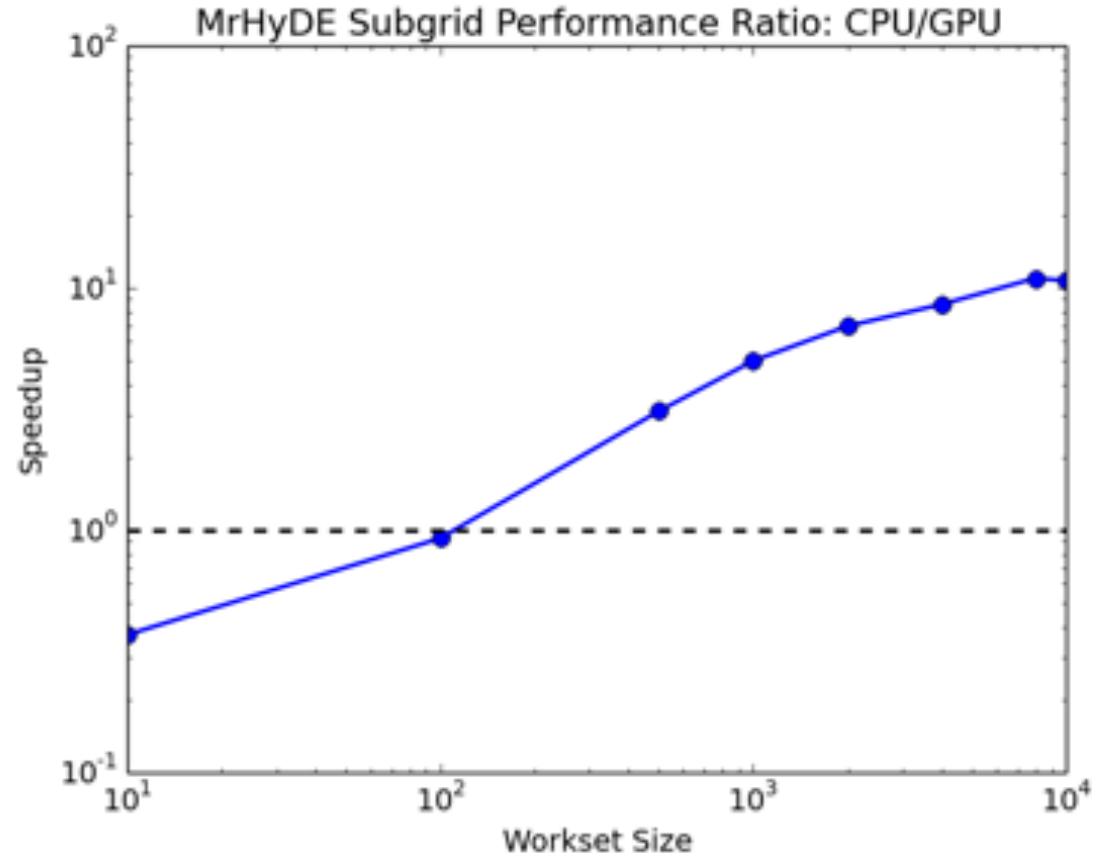
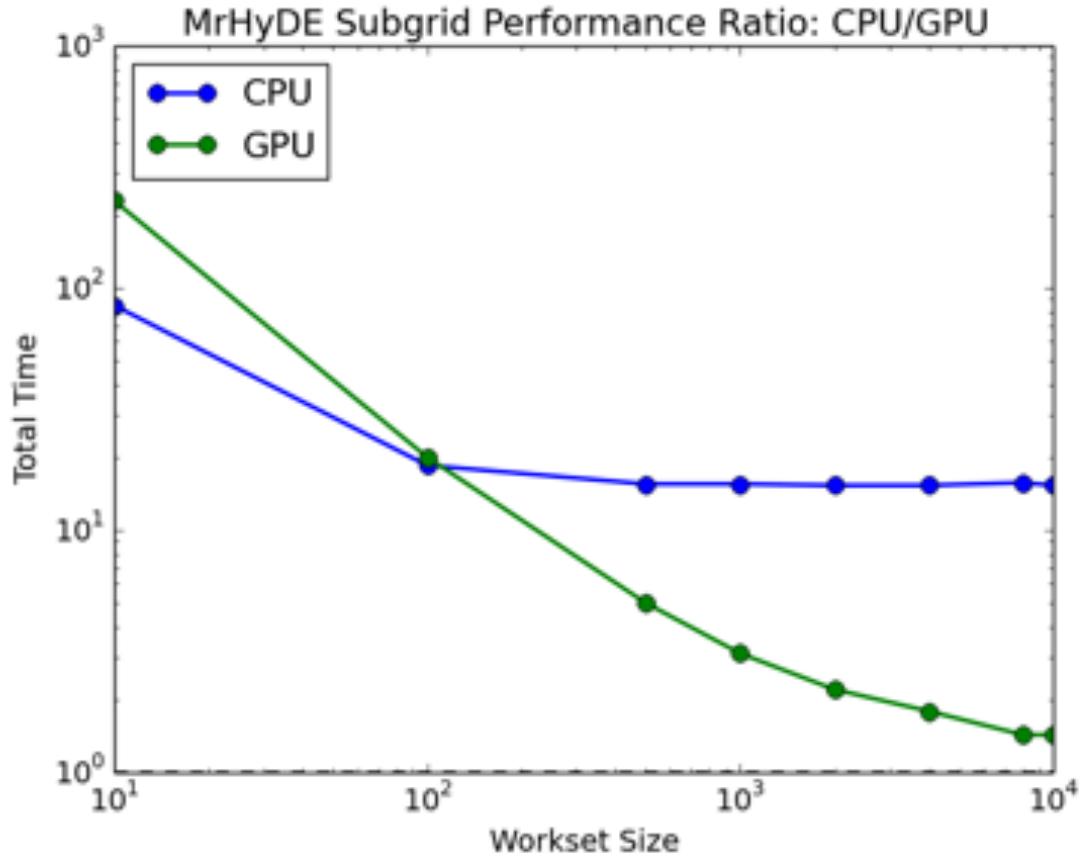
In this example, the linear solver uses the CPU, so the scatter needs to be on the CPU.

# MrHyDE Assembly: CPU vs GPU



If we also use the GPU for the linear solver, the scatter scales much better.  
Obtaining this much speed-up from the solver is another story.

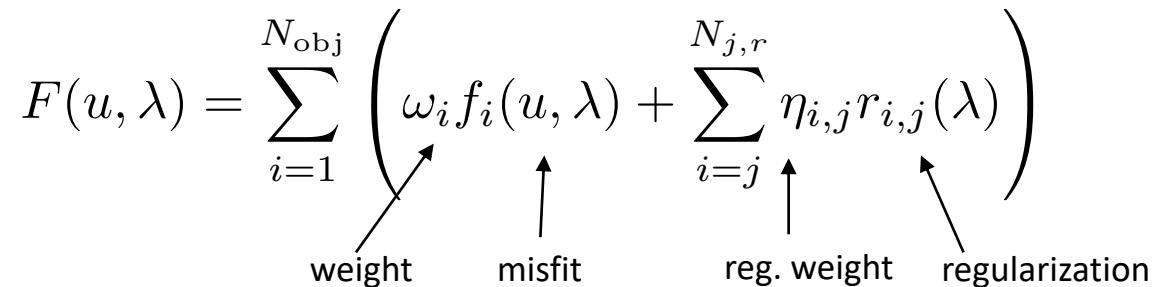
# MrHyDE Subgrid: CPU vs GPU



# **Large-scale PDE Constrained Optimization**

# Large-scale PDE Constrained Optimization

- MrHyDE has a fairly basic ROL interface that only requires objective values and gradients
  - An upgrade to the new version of ROL will occur by May 2021
- Parameters are defined through an optional block in the input file
- The function manager is aware of all of these, so use them like variables in functions
  - Syntax is slightly different if a parameter is a vector
- Support several different types of parameters:
  - Active: derivatives can be computed
  - Inactive: will not get updated during optimization/UQ
  - Discretized/distributed: discretization of a field, always active
  - Stochastic: meant for use with SOL, but hasn't been tested recently
- MrHyDE infrastructure abstracts the adjoint procedure, i.e., it is automatically enabled
  - multi-stage time integration has not been enabled yet
- General support for objective functions of the form

$$F(u, \lambda) = \sum_{i=1}^{N_{\text{obj}}} \left( \omega_i f_i(u, \lambda) + \sum_{j=j}^{N_{j,r}} \eta_{i,j} r_{i,j}(\lambda) \right)$$


# Various Options for Objective Functions

- Integrated control

$$f_i(u, \lambda) = \int_{\Omega} (r(u, \lambda, x) - q(x))^2 \ dx, \quad q(x) = \text{target function}$$

- Integrated response

$$f_i(u, \lambda) = (\bar{r} - \bar{q})^2, \quad \bar{r} = \int_{\Omega} r(u, \lambda, x) \ dx, \quad \bar{q} = \text{target data}$$

- Discrete control

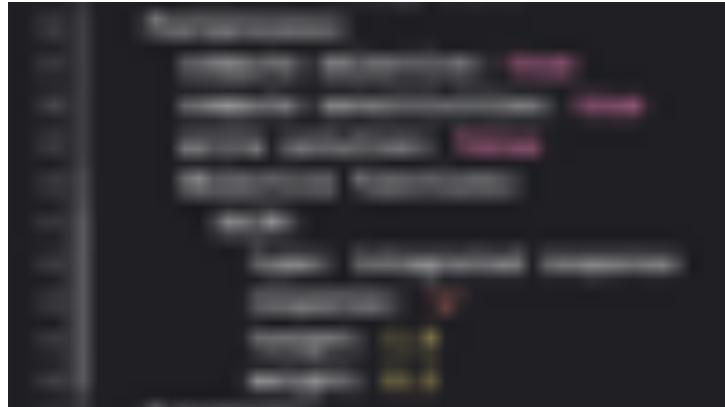
$$f_i(u, \lambda) = \|\mathbf{u} - \mathbf{q}\|_2^2, \quad \mathbf{u} = \text{discrete solution vector}, \quad \mathbf{q} = \text{discrete target vector}$$

- Sensor response

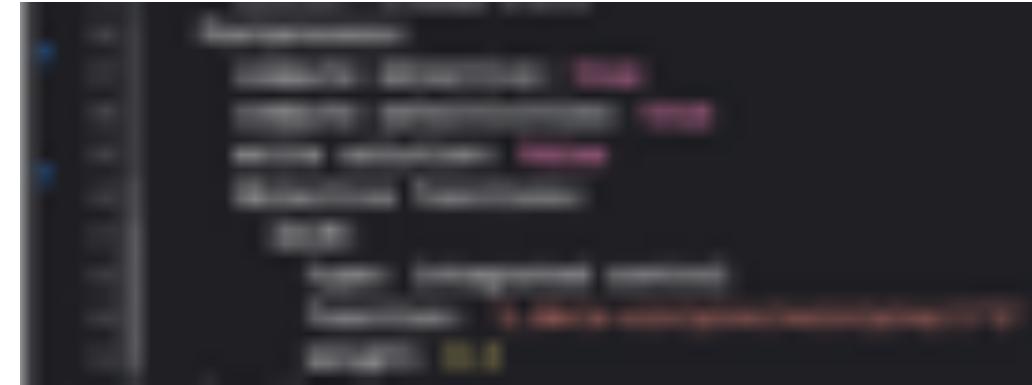
$$f_i(u, \lambda) = \sum_{j=1}^{N_{\text{sens}}} (r_j - q_j)^2, \quad r_j = r(u, \lambda, x_j), \quad x_j = \text{sensor location}, \quad q_j = \text{sensor data}$$

- User defines arbitrary number of objective and regularization functions in the input file
- Right now, these are scalarized into one objective to minimize
- Coming soon: generalization to allow for multi-objectives with separate gradients

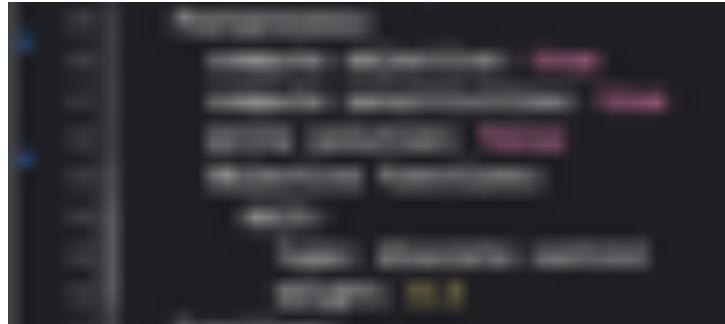
# Various Options for Objective Functions



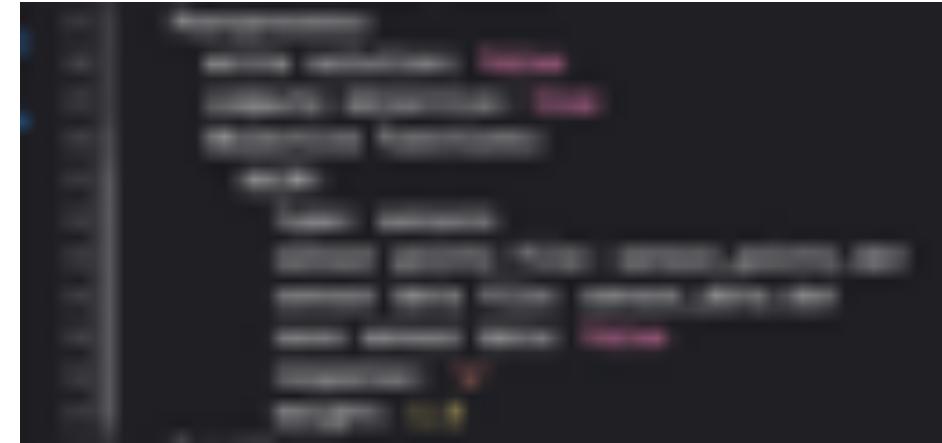
Integrated response



Integrated control



Discrete control



Sensor response

# Various Options for Objective Functions



A screenshot of a MATLAB code editor window. The code displays several objective function definitions, each containing regularization terms. The objectives are labeled `obj1`, `obj2`, `obj3`, `obj4`, and `obj5`. Each objective is defined as a sum of a data-fidelity term and a regularization term involving matrix `A` and vector `b`.

```
function obj = obj1(x)
    % ... (code for obj1)
end

function obj = obj2(x)
    % ... (code for obj2)
end

function obj = obj3(x)
    % ... (code for obj3)
end

function obj = obj4(x)
    % ... (code for obj4)
end

function obj = obj5(x)
    % ... (code for obj5)
end
```

Multiple objectives with regularization.



A screenshot of a MATLAB code editor window. The code shows multiple objective functions, each of which depends on a gradient calculation. The objectives are labeled `obj1`, `obj2`, `obj3`, `obj4`, and `obj5`. Each objective involves a call to a function `grad` to compute the gradient, followed by a summation of data-fidelity and regularization terms.

```
function obj = obj1(x)
    grad(x);
    % ... (code for obj1)
end

function obj = obj2(x)
    grad(x);
    % ... (code for obj2)
end

function obj = obj3(x)
    grad(x);
    % ... (code for obj3)
end

function obj = obj4(x)
    grad(x);
    % ... (code for obj4)
end

function obj = obj5(x)
    grad(x);
    % ... (code for obj5)
end
```

Multiple objectives that depend on gradients.

# Summary

## Day 1 - Introduction to Trilinos

- High-level overview of Trilinos
  - *An appropriate build of Trilinos will be available for anyone on the HPC systems. We will not be building Trilinos in this session. If someone does not have access to the HPC systems, I will work with them beforehand to get a build of Trilinos on their Mac or Linux machine.*
- Deeper dive into Kokkos and Sacado.
  - *A basic understanding of these packages will be helpful for day 2.*
- Exercise: creating and working with arrays (Kokkos Views) and automatic differentiation objects (Sacado AD)

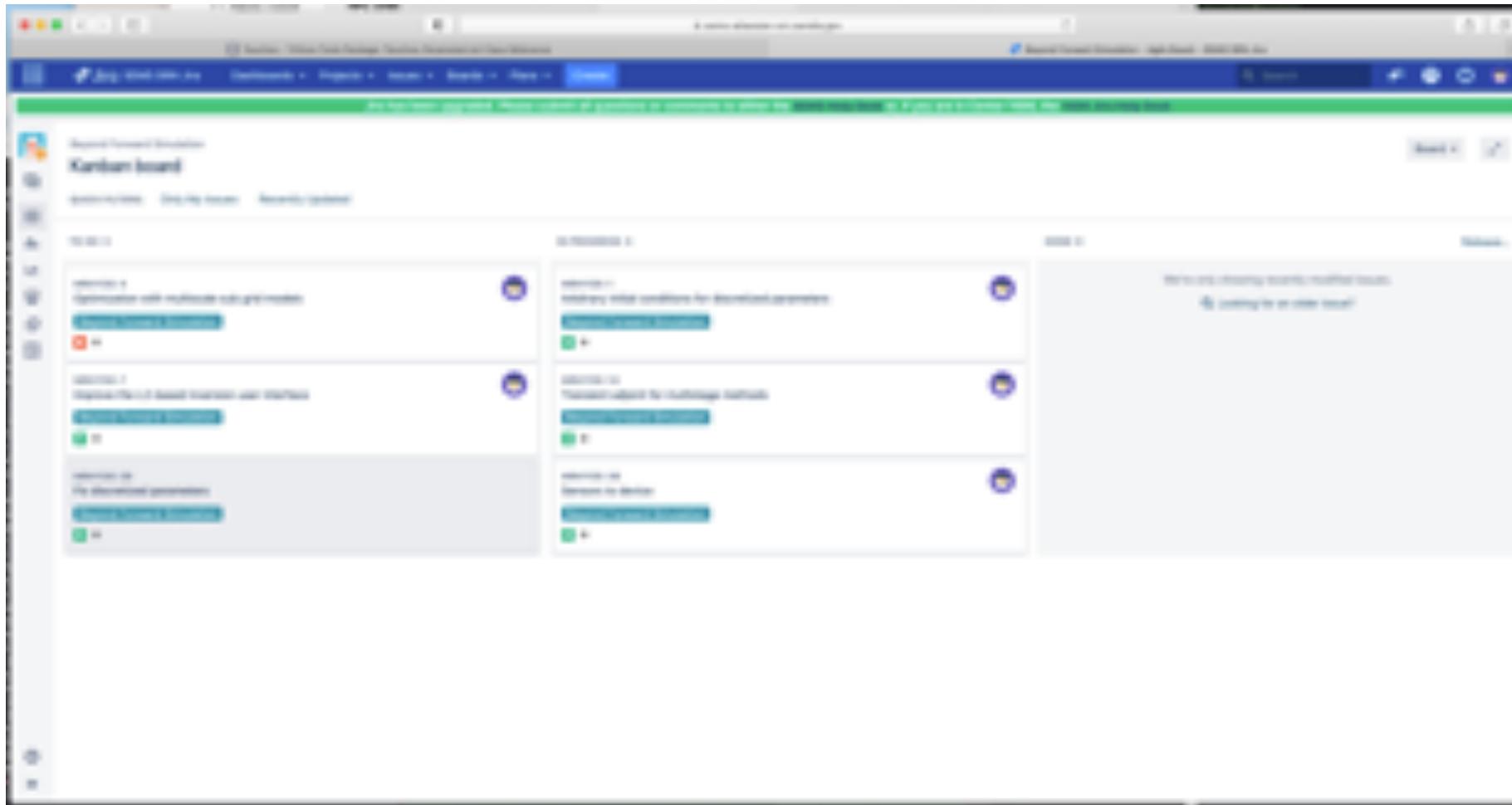
## Day 2 - Introduction to MrHyDE

- High-level overview of MrHyDE
- How to download, compile, run and visualize results
- Exercise: adding a new PDE in MrHyDE

## Day 3 - More advanced features in Trilinos/MrHyDE

- Hybridized methods and concurrent multiscale modeling
- Solving coupled multiphysics problems
- Performance portability and using heterogeneous computational architectures
- Large-scale PDE constrained optimization

# SEMS Jira Software Management



**Thank you for attending!**