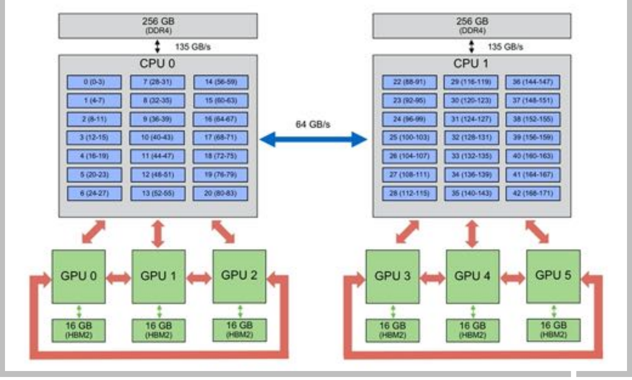
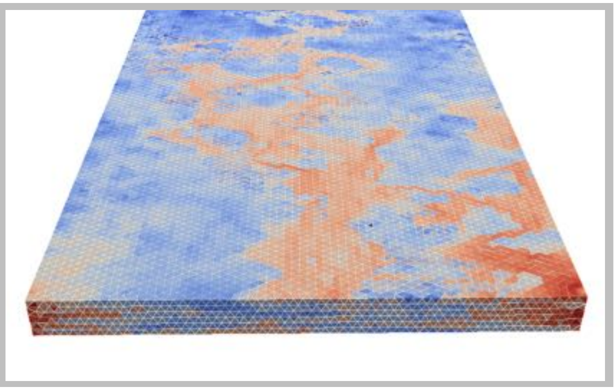
# Introduction to Trilinos and MrHyDE

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




**Tim Wildey**

**Optimization and Uncertainty Quantification Department**

**Center for Computing Research**



U.S. DEPARTMENT OF ENERGY | Office of Science

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

# 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**

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

# 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

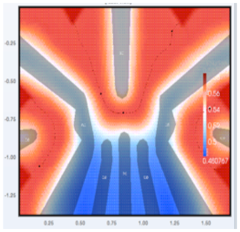
**Day 3 - More advanced features in Trilinos/MrHyDE**

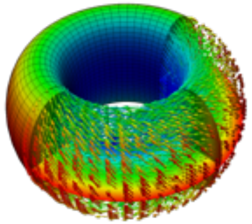
- Solving coupled multiphysics problems
- Performance portability and using heterogeneous computational architectures
- Large-scale PDE constrained optimization
- Concurrent multiscale modeling

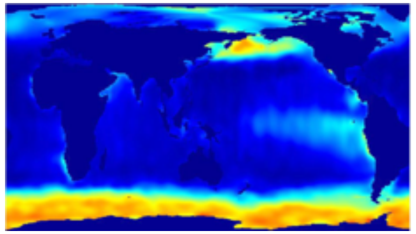
# Center for Computing Research

Leading Edge Algorithms and Enabling Technologies

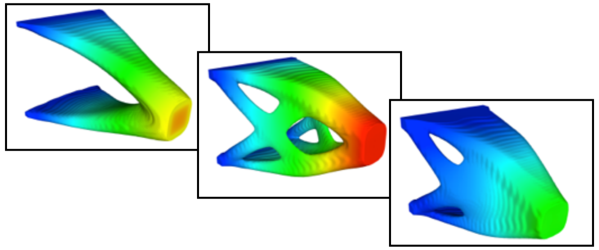
Strong External Collaborations

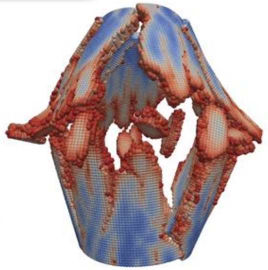





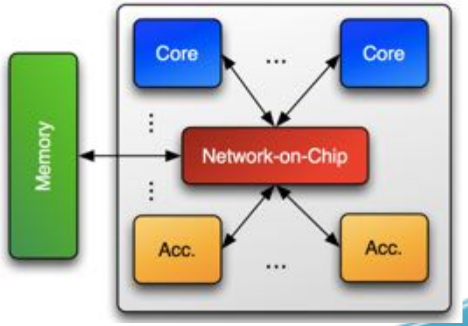


State-of-the-art Computational Science Applications



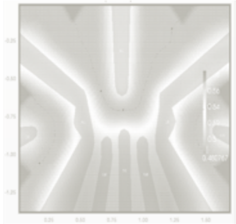


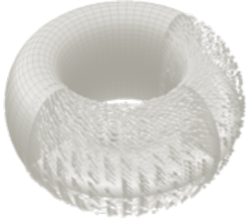
Scalable HPC Architecture and Systems Research



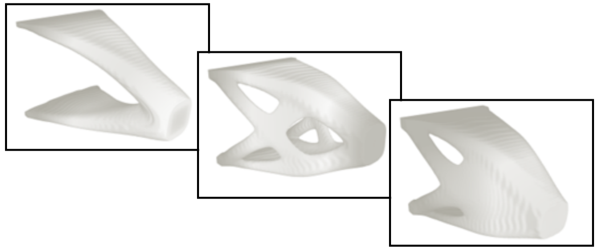
# Center for Computing Research

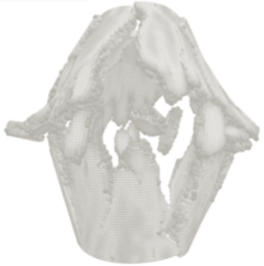
Leading Edge Algorithms and Enabling Technologies



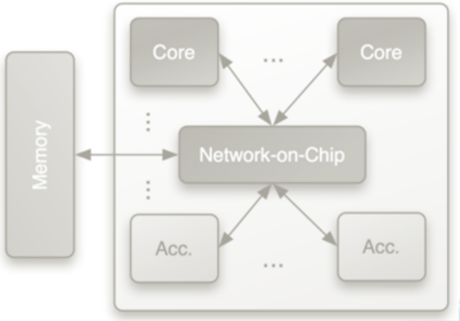


State-of-the-art Computational Science Applications





Scalable HPC Architecture and Systems Research




Strong External Collaborations


CCR Center for Computing Research

Sandia National Laboratories

# Trilinos is a Collection of Packages

| | Objective | Package(s) |
|---|---|---|
| Discretizations | Meshing & Discretizations | Intrepid, Pamgen, Sundance, Mesquite, STKMesh |
| | Time Integration | Rythmos |
| Methods | Automatic Differentiation | Sacado |
| | Mortar Methods | Moertel |
| Services | Linear algebra objects | Epetra, Tpetra |
| | Interfaces | Xpetra, Thyra, Stratimikos, Piro, … |
| | Load Balancing | Zoltan, Isorropia, Zoltan2 |
| | “Skins” | PyTrilinos, WebTrilinos, ForTrilinos, CTrilinos |
| | Utilities, I/O, thread API | Teuchos, EpetraExt, Kokkos, Phalanx, Trios, … |
| Solvers | Iterative linear solvers | AztecOO, Belos, Komplex |
| | Direct sparse linear solvers | Amesos, Amesos2, ShyLU (KLU2, Basker) |
| | Direct dense linear solvers | Epetra, Teuchos, Pliris |
| | Iterative eigenvalue solvers | Anasazi |
| | Incomplete factorizations | AztecOO, Ifpack, Ifpack2 |
| | Multilevel preconditioners | ML, CLAPS, MueLu |
| | Block preconditioners | Meros, Teko |
| | Nonlinear solvers | NOX, LOCA |
| | Optimization | MOOCHO, Aristos, TriKota, GlobiPack, OptiPack, ROL |
| | Stochastic PDEs | Stokhos |

# Trilinos – "A string of pearls"

The original design. This is how Trilinos looks to experienced developers.

Unfortunately, this is how Trilinos appears to many new users.

# MrHyDE is Built Upon a Small Subset of These Packages

| Trilinos Package | Usage in MrHyDE |
|---|---|
| **Teuchos** | smart pointers, parameter lists, MPI communicators, timers |
| **Panzer-STK** | interfacing with mesh data structures (exodusII) |
| **Intrepid2** | discretization tools |
| **Panzer-DOFManager** | keeping track of degree of freedom numbering in a multi-physics, multi-block setting with physics-compatible discretizations |
| **Sacado** | automatic differentiation |
| **Kokkos** | performance portability |
| **Tpetra** | linear algebra stuff, e.g., vectors and matrices |
| **Belos** | iterative linear solvers (CG, GMRES) for distributed sparse linear systems |
| **MueLu** | preconditioners (mainly AMG) for linear systems |
| **Amesos2** | direct linear solvers |
| **ROL** | large-scale PDE constrained optimization |

# Obtaining MrHyDE and the Exercises

**If you are on SRN HPC (chama, skybridge), grab an interactive node**

```
salloc -N1 --time=1:00:00 --account=FY210060 --partition=short,batch
```

**Download using git:**

```
git clone https://github.com/TimWildey/MrHyDE.git
```

**If git fails and you are on SRN HPC:**

```
export https_proxy=http://wwwproxy.sandia.gov:80

git clone https://github.com/TimWildey/MrHyDE.git
```

**If git still fails and you are on SRN HPC:**

```
cp /projects/MrHyDE/backups/MrHyDE.tar .

tar -xvf MrHyDE.tar
```

# Setting up Your Environment

**<u>If you are on SRN HPC (chama, skybridge):</u>**

```
source MrHyDE/scripts/load-gcc-env
```

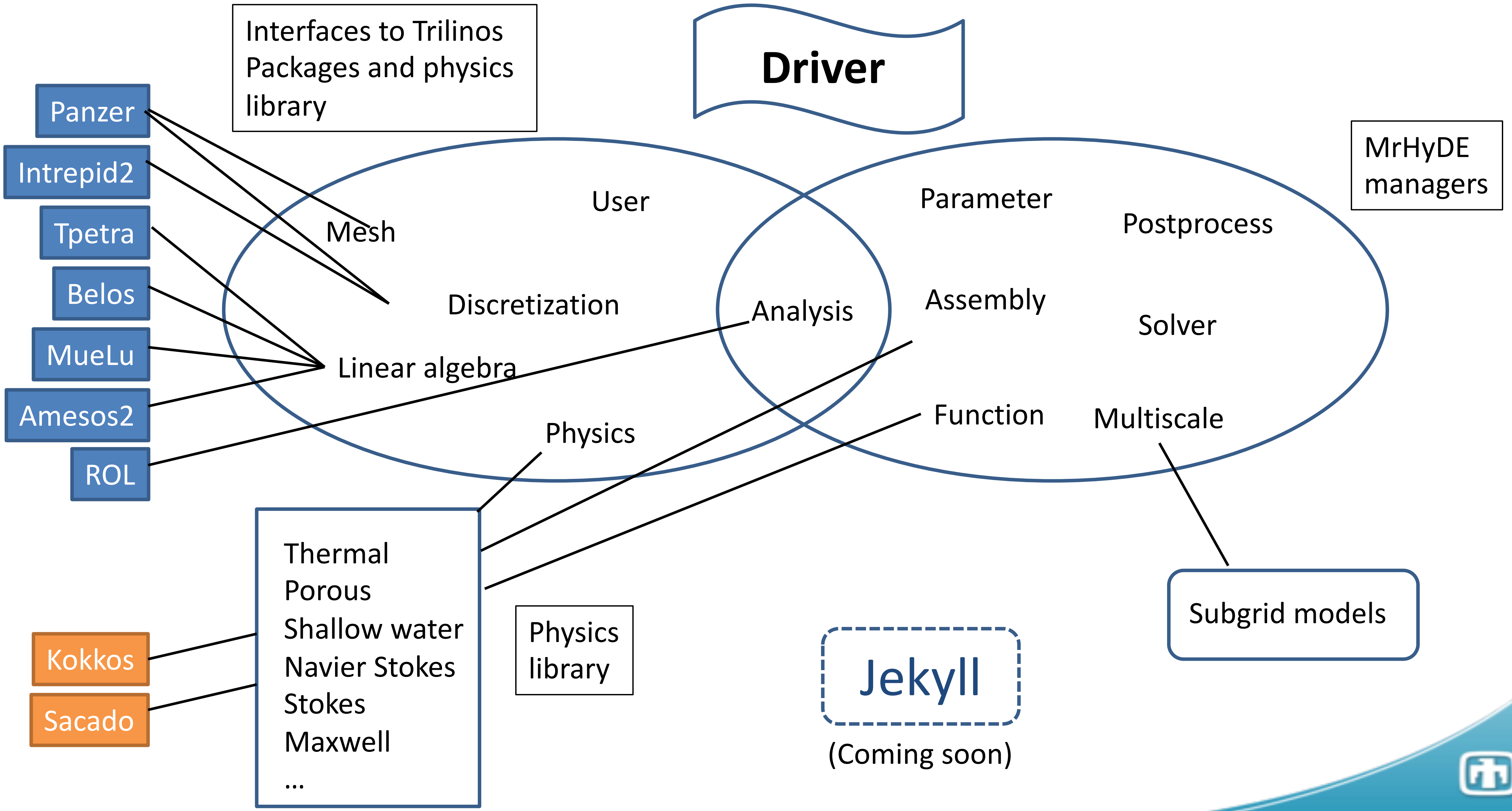
**<u>If you are on your own machine and you already built Trilinos yourself:</u>**

Your environment should already be setup or you probably know how to do this.

**<u>If you are on your own machine and you have not built Trilinos:</u>**

You'll need to build Trilinos after this first session. Email me if you want help.

# MrHyDE Organization

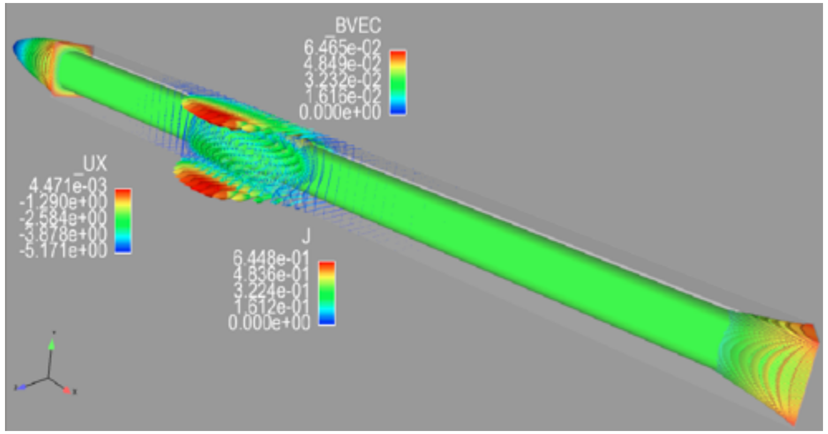


# What is Automatic Differentiation (AD)?

- Sounds fancy, but really it's completely straightforward
- Not the same as symbolic differentiation (no functional forms) or numeric differentiation (finite differences)
- Many languages/packages have some form of AD
  – Trilinos uses Sacado
- Object-oriented programming makes this much easier
- Writing an AD package is easy - all we need is basic calculus and the concept of operator overloading.
  – I wrote a MATLAB AD package for ACES
- Operator overloading – redefining how an operator works for different inputs

Why are derivatives needed?

- Accurate Jacobians
- Optimization
- Sensitivity analysis
- UQ
- Stability analysis



Iso-velocity adjoint surface for fluid flow in a 3D steady MHD generator in Drekar computed via Sacado

# How Does Automatic Differentiation Work?

Let $x \in \mathbb{R}$ and let $f(x)$ and $g(x)$ be functions of $x$

Define a new function: $h(x) = f(x)g(x)$

For a given value of $x$, suppose we know $f(x), g(x), \frac{df}{dx}, \frac{dg}{dx}$

Can we compute $h(x)$ and $\frac{dh}{dx}$? Of course.

$$h(x) = f(x)g(x), \qquad \frac{dh}{dx} = \frac{df}{dx}g(x) + f(x)\frac{dg}{dx}$$

Sacado works by creating a class that stores both the values and derivatives

All mathematical operators (+,*,sin,exp, ...) are overloaded for this class

Writing `h = f * g` really computes

```
h.val = f.val * g.val
h.der = f.der * g.val + f.val * g.der
```

# Building the Exercises

- Go into MrHyDE/doc/Tutorial/
- We will be building both exercises just like we will build MrHyDE
- The exercises are in MrHyDE/doc/Tutorial/Exercises, but we will use a separate build directory to compile
- Go into MrHyDE/doc/Tutorial/build
- Open the configure file and edit

```
TRILINOS_HOME=‘/projects/MrHyDE/Trilinos'
TRILINOS_INSTALL=‘/projects/MrHyDE/Trilinos/install-gnu-8.2.1’
EXERCISES_HOME=‘/nscratch/USERNAME/Software/MrHyDE/doc/Tutorial/Exercises'
```

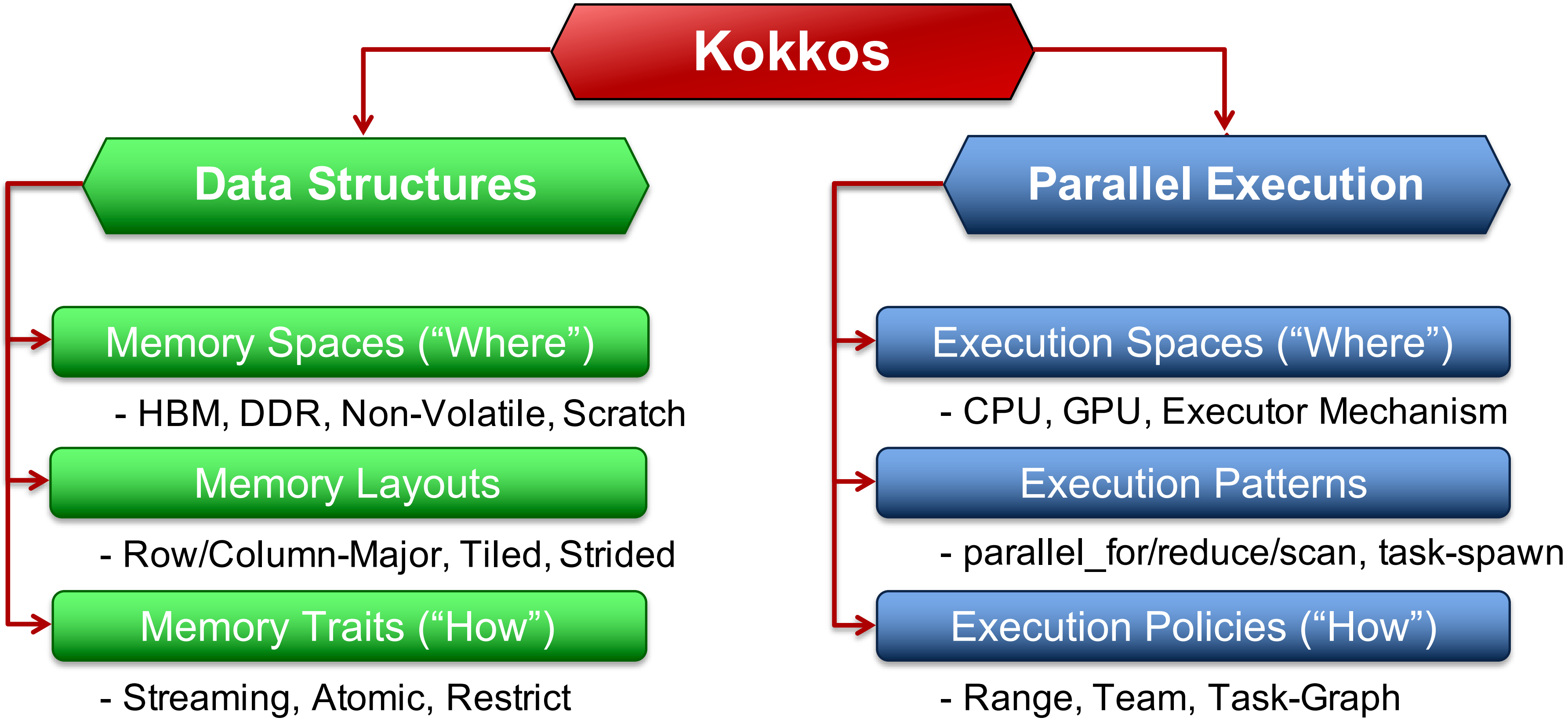
to point to your Trilinos build and your MrHyDE directory

- Then, run `./configure`
- Then run `ninja`

# Exercise with AD

- The first exercise explores using AD objects.
- The file is Exercise_AutoDiff.cpp and it creates an executable in the build directory called autodiff.
- Take ~10 minutes to play with the exercise, i.e., run `./autodiff`
- Make sure you understand what it is doing.
- Please feel free to ask questions
- Modify it with different functions and rerun `ninja` in the build directory.
- Can you break it?
- If you are feeling ambitious, compare with a finite difference approximation

# Kokkos Library for Performance Portability



# Multi-dimensional Arrays

- All codes require some type of multi-dimensional arrays
- Desirable properties:
  - Easy to use syntax
  - Supports at least 3-dimensional arrays, ideally higher
  - Can contain a wide variety of objects, e.g., arrays of doubles or AD objects
  - Allow user to control where the memory is allocated/accessed (CPU, GPU, UVM, …)
  - No hidden copies of data
  - Data layouts optimal for where data is needed
- Most of Trilinos has transitioned to using specialized Kokkos arrays, called Views.

**View** abstraction

- A *lightweight* C++ class with a pointer to array data and a little meta-data,
- that is *templated* on the data type (and other things).

# Simple Usage of Views

**View** overview:

- **Multi-dimensional array** of 0 or more dimensions
  scalar (0), vector (1), matrix (2), etc.
- **Number of dimensions (rank)** is fixed at compile-time.
- Arrays are **rectangular**, not ragged.
- **Sizes of dimensions** set at compile-time or runtime.
  e.g., 2x20, 50x50, etc.
- Access elements via "(...)" operator.

**Example**:

```
View<double***> data("label", N0, N1, N2); //3 run, 0 compile
View<double**[N2]> data("label", N0, N1);  //2 run, 1 compile
View<double*[N1][N2]> data("label", N0);   //1 run, 2 compile
View<double[N0][N1][N2]> data("label");    //0 run, 3 compile
//Access
data(i,j,k) = 5.3;
```

Note: runtime-sized dimensions must come first.

# Simple Usage of Views

**View** life cycle:

- Allocations only happen when *explicitly* specified.
  i.e., there are **no hidden allocations**.
- Copy construction and assignment are **shallow** (like pointers).
  so, you pass `Views` by value, *not* by reference
- Reference counting is used for **automatic deallocation.**
- They behave like `shared_ptr`

# Simple Usage of Views

**View** life cycle:

- Allocations only happen when *explicitly* specified.
  i.e., there are **no hidden allocations**.
- Copy construction and assignment are **shallow** (like pointers).
  so, you pass `Views` by value, *not* by reference
- Reference counting is used for **automatic deallocation.**
- They behave like `shared_ptr`

**Example**:

```
View<double*[5]> a("a", N0), b("b", N0);
a = b;
View<double**> c(b);
a(0,2) = 1;
b(0,2) = 2;
c(0,2) = 3;
print a(0,2)
```

What gets printed?

# Exercise Using Kokkos Views

- We are going to get our feet wet using Kokkos Views
  - On Wednesday, we'll learn why everything in this exercise is suboptimal
- The file is Exercise_KokkosViews.cpp and it creates an executable in the build directory called `views`
- Take ~10 minutes to build/run the executable, play with the code, recompile, and inspect the output.
- Please feel free to ask questions
- Can you figure out how to make a multi-dimensional array?
- How about an array of Sacado AD objects? Hint – add `#include "Sacado.hpp"`
  - 2-dimensional arrays of AD objects are critical in MrHyDE

```
residual(elem,dof) = dudx*dvdx + lambda*u*v - src*v;
```

Gives $J = \frac{\partial R}{\partial u}$ if $u$ is "seeded" and $\frac{\partial R}{\partial \lambda}$ if $\lambda$ is "seeded"

# Preview of Tomorrow

<u>Day 1 - Introduction to Trilinos</u>

- 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)

<u>Day 2 - Introduction to MrHyDE</u>

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

<u>Day 3 - More advanced features in Trilinos/MrHyDE</u>

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