Panzer: A Finite Element Assembly Engine for Multiphysics Simulation

Roger Pawlowski, Eric Cyr, and John Shadid
Sandia National Laboratories

Trilinos User Group Meeting
November 2nd, 2011

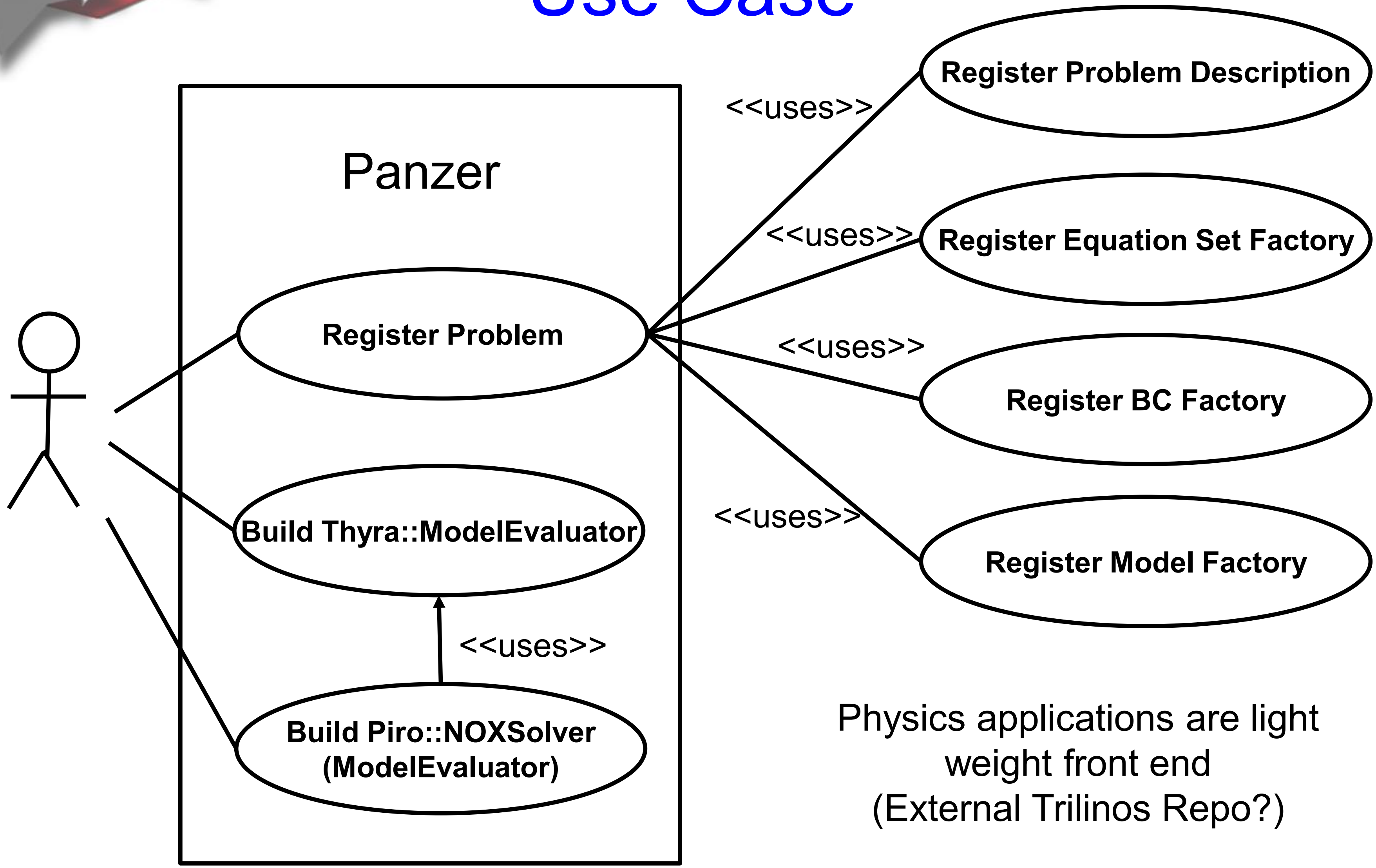
SAND2011-8261C

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

What is Panzer?

- A general finite element assembly engine for multiphysics simulation:
  - **User Physics Kernels + Problem Description = Thyra::ModelEvaluator**
    - Quantities need for advanced **solution** and **analysis** algorithms: residuals, Jacobians, parameter sensitivities, stochastic residual/Jacobians, etc.
  - A **unification** of Trilinos discretization tools: Shards, Intrepid, Phalanx, Sacado, Stokhos, (Optionally: STK, SEACAS)
  - Supports 1D, 2D, and 3D unstructured mesh calculations
- A library and a Trilinos package – NOT a terminal application
- Contains NO physics specific code
  - Generic assembly tools
- Leverages Template-based Generic Programming to assemble quantities of interest

Use Case



Physics applications are light weight front end (External Trilinos Repo?)

History

- Research over past 7 years in Charon:
  - Export control (4D001) restricted collaborations
  - Complicated build system (some great features including TPL management)
  - Restricted to a **Monolithic Framework**
  - No longer meets research requirements

- Generalize the capabilities explored and developed in Charon into Trilinos packages
  - **Rapid prototyping of new discretizations/algoritms**
  - New code base → flexibility, lessons learned
  - Resulting packages: Phalanx, Panzer

New Research Requirements

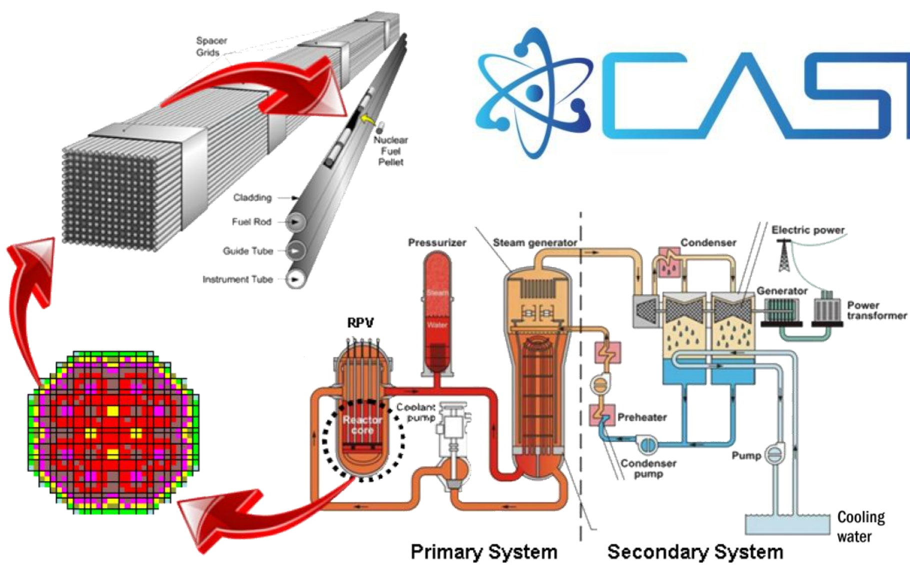
A Research Tool for DOE/OS: ASCR/AMR, ASCR/UQ

- **Formulations: fully coupled fully implicit, semi-implicit, FCT**
- **Compatible discretizations:**
  - Mixed basis for DOFs within element block
  - Arbitrary element types (not restricted to nodal basis)
  - “Node” specific code is eliminated (or treated as specializations)
- **Multiphysics:**
  - Fully coupled systems composed of different equation sets in different element blocks
  - Preconditioning: Approximate block factorization/physics based
- **Supports advanced analysis techniques:**
  - **Modern software techniques for advanced architectures**
  - Supports Template-based Generic Programming
  - Adjoint-based error analysis
  - Stability, bifurcation, embedded (SAND) optimization, embedded uncertainty quantification (Stokhos/PCE)

Production Requirements

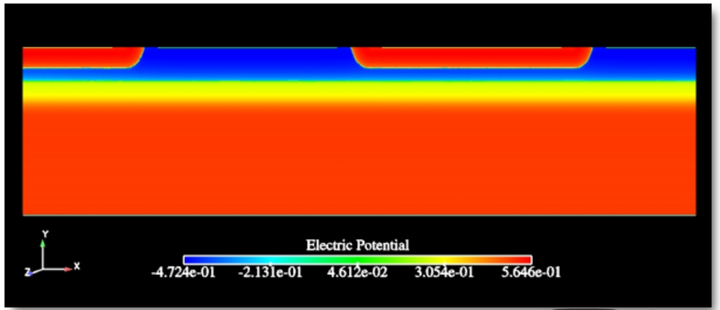
Production Quality Software (ASC, CASL)

- Strict and extensive unit testing (TDD)
- Integration with legacy code components
- NOT restricted to any mesh database or I/O format
- Control over granularity of assembly process (efficiency vs flexibility)
- Applications:
  - ASC: Semiconductor Device (Next-generation Charon) for QASPR
  - CASL: CFD component for VERA simulator



CASL

DOE / NNSA

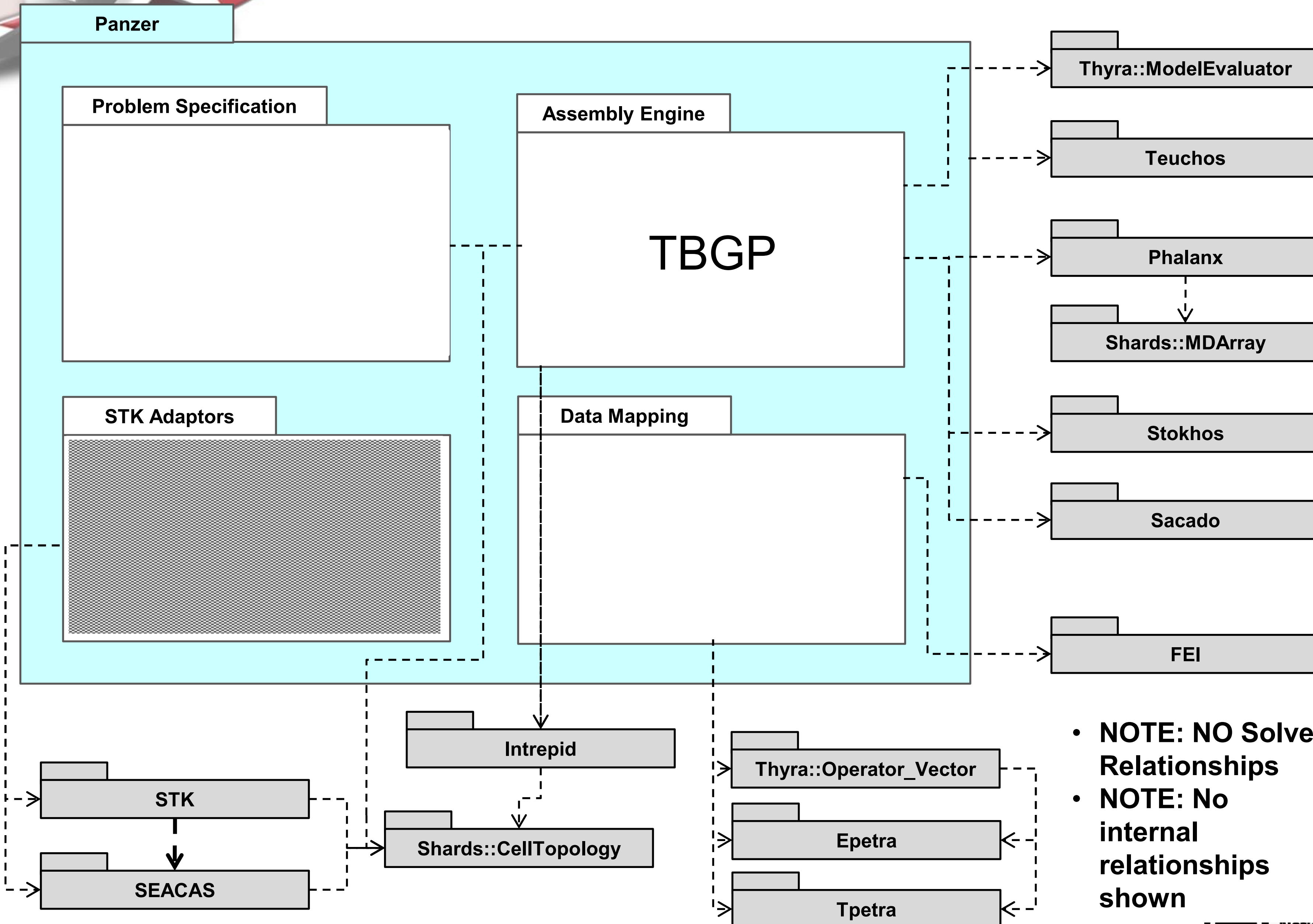


Sandia National Laboratories

Panzer Components

- Problem Description
  - Maps equations sets and boundary conditions into nodes of Phalanx assembly DAG.
- Assembly Engine
  - A collection of Phalanx Field Managers to control assembly
  - Produces a Model Evaluator for User
- Data Mapping Utilities
  - DOF Manager for mapping field values into linear algebra
  - Connection Manager: Abstraction of Mesh
- STK Adaptors (Optional)
  - Concrete implementation Panzer objects for using STK::Mesh and SEACAS for I/O
  - Specialized evaluators

Panzer Unifies Trilinos Discretization Tools



- NOTE: NO Solver Relationships
- NOTE: No internal relationships shown

Graph-based Assembly Process

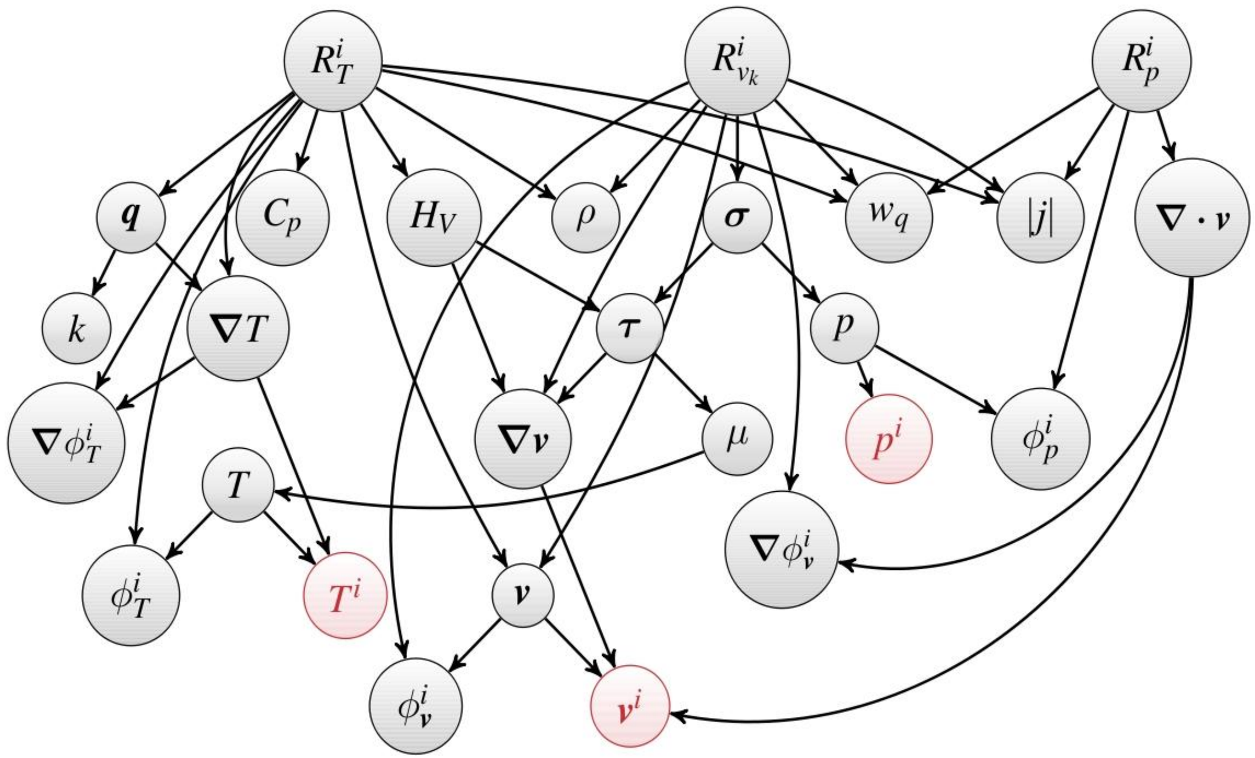
(Notz, Pawlowski, Sutherland; submitted to TOMS)

- Phalanx package
- Graph-based equation description
  - Automated dependency tracking (Topological sort to order the evaluations)
  - Each node is a point of extension that can be swapped out
  - Easy to add equations
  - Easy to change models
  - Easy to test in isolation
- Multiphysics Complexity is handled automatically!
- User controlled memory allocation of Field data
- Multi-core research:
  - Spatial decomposition (Kokkos::MDArray)
  - Algorithmic decomposition

$$R_T^i = \sum_{e=1}^{N_e} \sum_{q=1}^{N_q} \left[\left(\rho C_p \boldsymbol{v} \cdot \nabla T - H_V \right) \phi_T^i - \boldsymbol{q} \cdot \nabla \phi_T^i \right] w_q |j| = 0$$

$$R_{v_k}^i = \sum_{e=1}^{N_e} \sum_{q=1}^{N_q} \left[\rho \boldsymbol{v} \cdot \nabla \boldsymbol{v} \phi_{\boldsymbol{v}}^i + \boldsymbol{\sigma} : \nabla \left(\phi_{\boldsymbol{v}}^i \boldsymbol{e}_k \right) \right] w_q |j| = 0$$

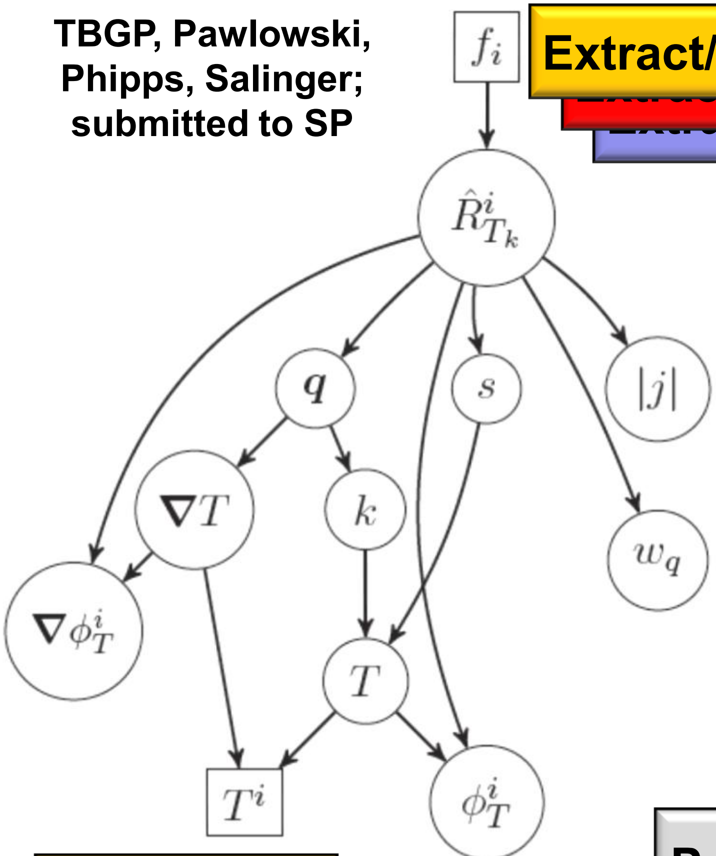
$$R_p^i = \sum_{e=1}^{N_e} \sum_{q=1}^{N_q} \nabla \cdot \boldsymbol{v} \phi_p^i w_q |j| = 0$$



Phalanx Handles Multiphysics Complexity using Template-based Generic Programming

TBGP, Pawlowski, Phipps, Salinger; submitted to SP

Extract/Scatter



Gather/Seed

$$f(x) = \sum_{k=1}^{N_w} f_k = \sum_{k=1}^{N_w} Q_k^T \hat{R}^i_{T_k}(P_k x)$$

$$\hat{R}^i_T = \sum_{e=1}^{N_e} \sum_{q=1}^{N_q} \left[-\nabla \phi^i_T \cdot \mathbf{q} + \phi^i_T s\right] w_q |j| = 0$$

Evaluation Type	Scalar Type
$f(x,p)$	`double`
$J = \frac{\partial f}{\partial x}$	`DFad<double>`
$\frac{\partial^2 f}{\partial x_i \partial x_j}$	`DFad< DFad<double> >`

Param. Sens., Jv, Adjoint, PCE (SGF, SGJ), Arb. Prec.

`PCE::OrthogPoly<double>`

`DFad<PCE::OrthogPoly<double> >`

Take Home Message:
Reuse the same code base!
Equations decoupled from algorithms!
Machine precision accuracy!

Sandia National Laboratories

Data Mapping

Computes global unknown indices

1. Serves as interface to mesh
2. Allows Panzer to be mesh agnostic
3. Handles unknowns for mixed discretizations
4. Handles unknowns for multiphysics (multiple element blocks)
5. Uses FEI for producing unknowns

Composed of 3 primary pieces

1. FieldPattern – Describes the basis layout and continuity of fields
2. DOFManager – Manages and computes unknown numbers on fields
3. ConnManager – (User implemented) Mesh topology from field pattern

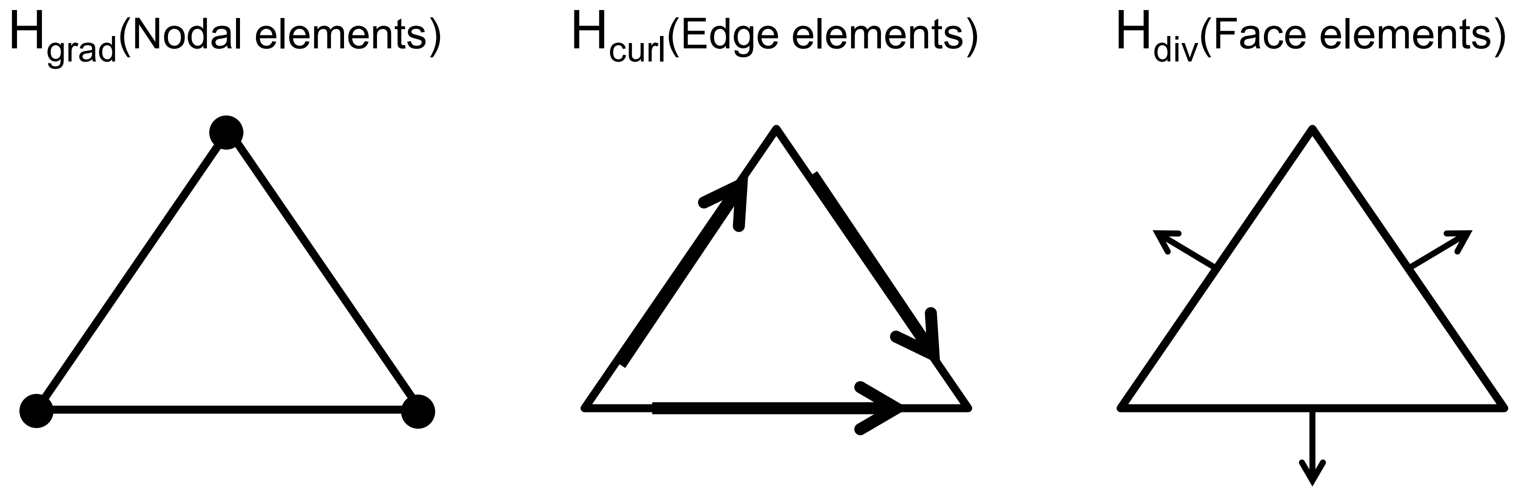
Features not implemented but supported by design

1. Higher order discretizations – geometric symmetries
2. Heterogeneous meshes – quadrilaterals and triangles

Data Mapping: New Directions

Finite Element discretizations have changed

- Charon used nodal-equal-order-finite elements
- New code embraces mixed discretizations
- Also using “Compatible Discretizations”
- Requires extra data management: orientations

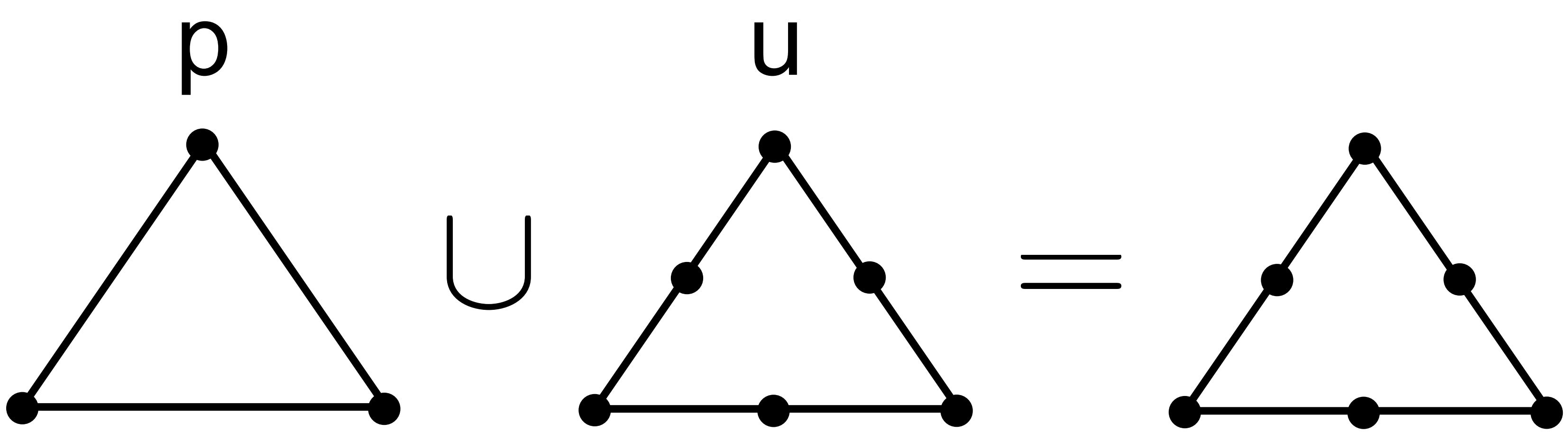


Data Mapping Handles These Elements

Data Mapping: Field Pattern

For stable Navier-Stokes pair:

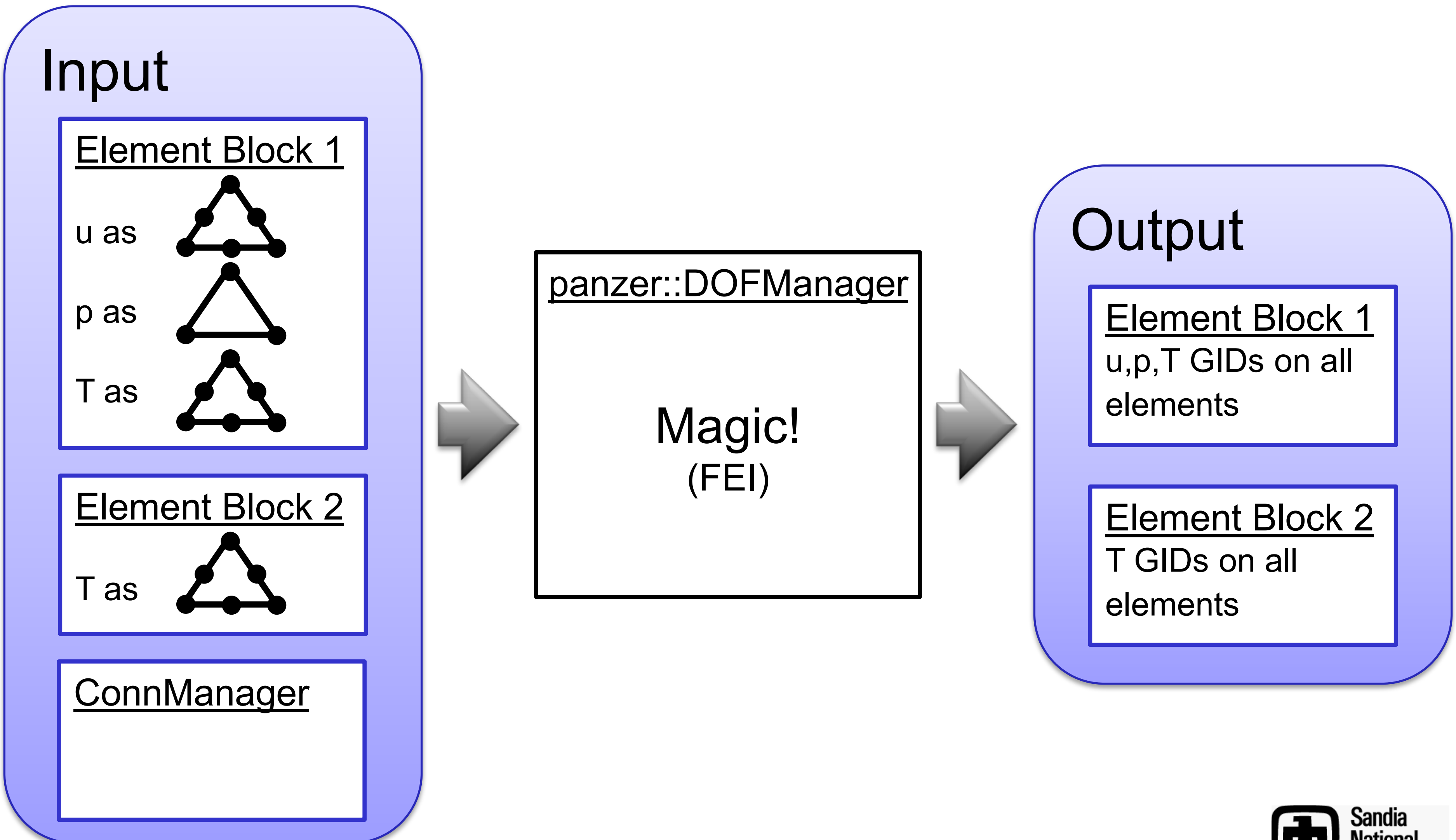
- Linear pressures
- Quadratic velocities



Field Pattern specifies basis layout

- Continuity across subcells (continuity of field)
- Unknowns on each element
- Communicates required topology

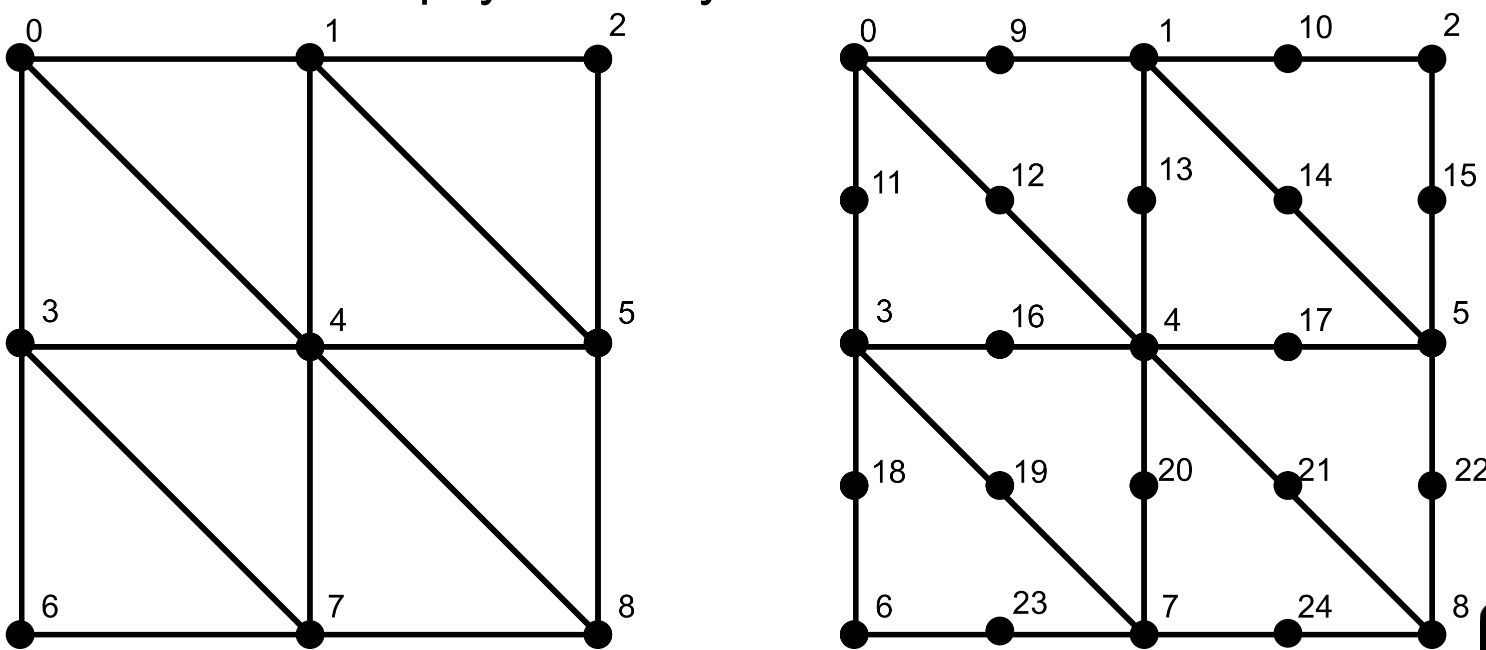
Data Mapping: DOFManager



Data Mapping: ConnManager

Must generate mesh connectivity

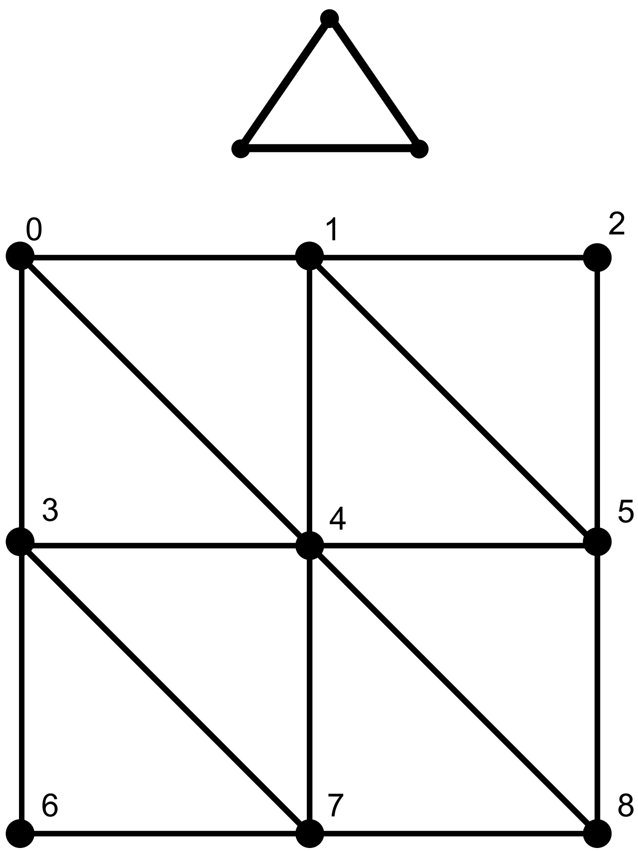
- DOFManager passes in field pattern
- Provides unique global node, edge, volume ids for each element
- Optionally provides orientation for edge and face elements
- Uniform field pattern across all element blocks
  - Makes multiphysics easy



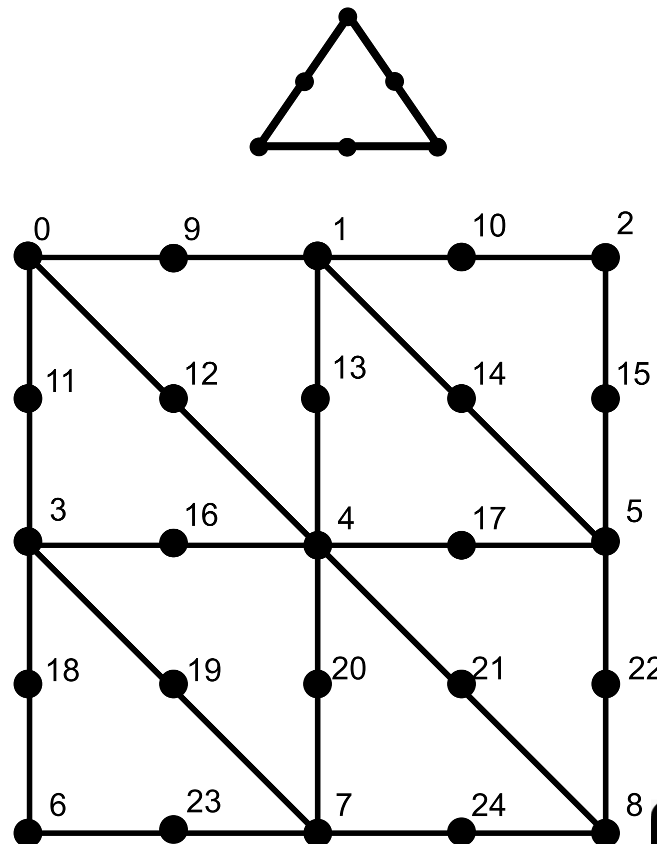
Sandia National Laboratories

Data Mapping: ConnManager

Piecewise linear p
Piecewise linear u



Piecewise linear p
Piecwise quadratic u



Data Mapping: Unknown Ordering

Old code used "interlaced" unknown ordering by node

$$[u_0, v_0, p_0, u_1, v_1, p_1, u_2, v_2, p_2, \ldots, u_N, v_N, p_N]^T$$

Panzer data mapping allows for greater control of ordering

- You can still interlace (the default)
- Blocked physics is also possible

Same ConnManager can be used multiple times

- Produce DOFManager for each type of physics
- Good for Block Preconditioning

$$\overset{u_0 v_0 p_0 \ldots u_8 v_8 p_8}{[A]} \longrightarrow \begin{bmatrix} \overset{u_0 v_0 \ldots u_8 v_8}{F} & \overset{p_0 \ldots p_8}{B^T} \\ B & C \end{bmatrix}$$

The Future

- Stokhos integration (almost complete)
- Adjoint capability
- Use of Kokkos MDArray for multi-/many-core/GPGPU support
- Expression templates for MDFields
- Phalanx: Incorporation of Kokkos::MDArray (Evaluators will be functors)