SAND2016-10689C

An Overview of Trilinos

Mark Hoemmen & Alicia Klinvex
Sandia National Laboratories
24 Oct 2016

Sandia is a multiprogram 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.

Sandia National Laboratories

Outline

- What can Trilinos do for you?
- Trilinos' software organization
- Whirlwind tour of Trilinos packages

What can Trilinos do for you?

What is Trilinos?

- Object-oriented software framework for…
- Solving big complex science & engineering problems
- More like LEGO™ bricks than Matlab™

Applications

- All kinds of physical simulations:
  - Structural mechanics (statics & dynamics)
  - Circuit simulations (physical models)
  - Electromagnetics, plasmas, & superconductors
  - Combustion & fluid flow (at macro- & nanoscales)
- Coupled / multiphysics models
- Data and graph analysis
  - Even gaming!

Target platforms:
Any and all, current and future

- Laptops & workstations
- Clusters & supercomputers
  - Multicore CPU nodes
  - Hybrid CPU / GPU nodes
- Parallel programming environments
  - MPI, OpenMP, Pthreads, …
  - CUDA (for NVIDIA GPUs)
  - Combinations of the above
- User “skins”
  - C++ (primary language)
  - Python (PyTrilinos)
  - Web (Hands-on demo)

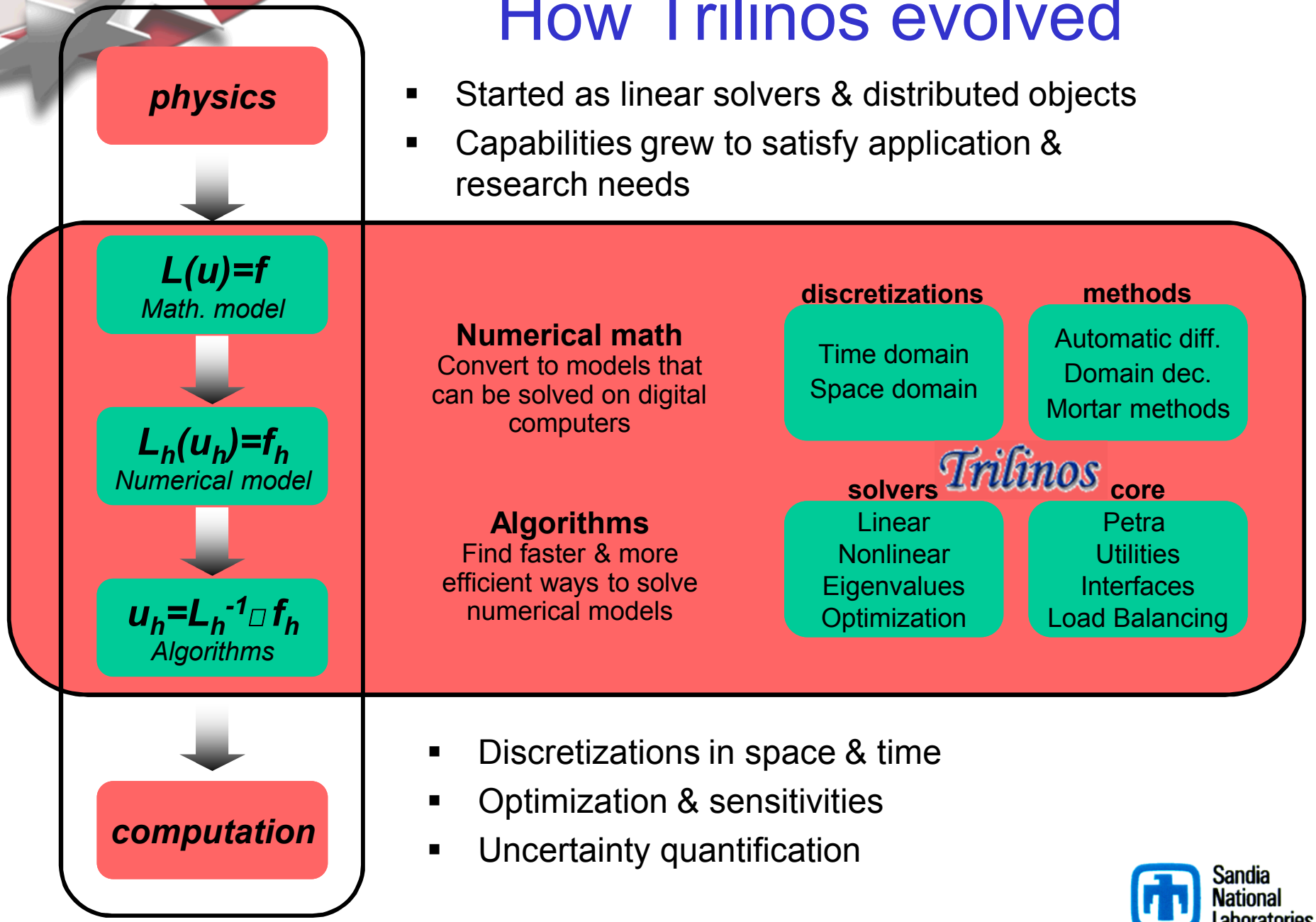
Sandia National Laboratories

Unique features of Trilinos

- Huge library of algorithms
  - Linear & nonlinear solvers, preconditioners, …
  - Optimization, transients, sensitivities, uncertainty, …
  - Discretizations, mesh tools, automatic differentiation, …
- Package-based architecture
- Support for huge (> 2B unknowns) problems
- Support for mixed & arbitrary precisions
- Growing support for hybrid (MPI+X) parallelism
  - X: Threads (CPU, Intel Xeon Phi, CUDA on GPU)
  - Built on Kokkos
    - Shared-memory parallel programming model
    - Write code once, run it on CPUs or GPUs or …

How Trilinos evolved

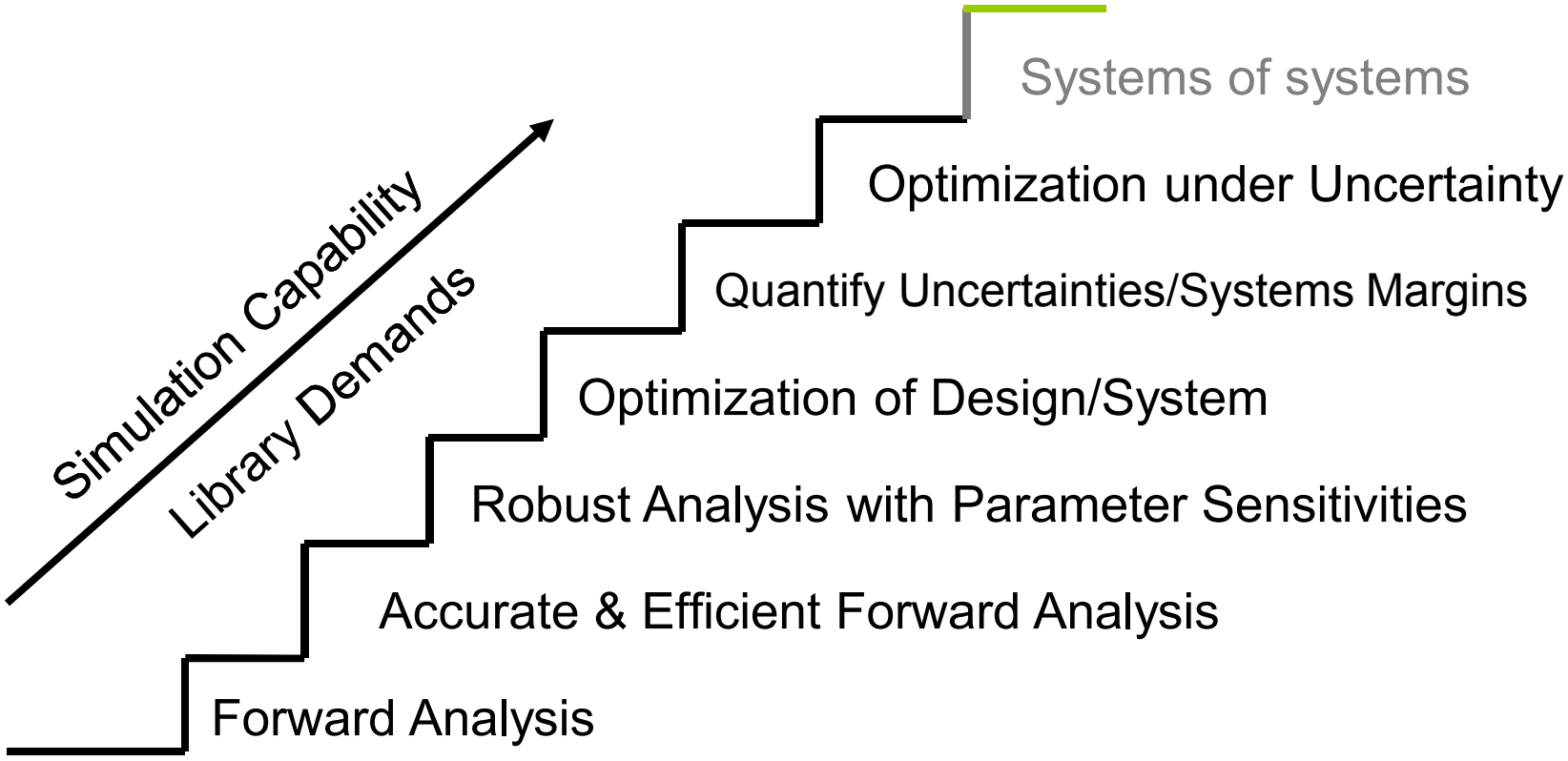
- Started as linear solvers & distributed objects
- Capabilities grew to satisfy application & research needs



- Discretizations in space & time
- Optimization & sensitivities
- Uncertainty quantification

Sandia National Laboratories

From Forward Analysis, to Support for High-Consequence Decisions



Each stage requires *greater performance* and *error control* of prior stages:
Always will need: more accurate and scalable methods.
more sophisticated tools.

Trilinos strategic goals

- Algorithmic goals
  - *Scalable* computations (at all levels of parallelism)
  - *Hardened* computations
    - Fail only if problem intractable
    - Diagnose failures & inform the user
  - Full vertical coverage
    - Problem construction, solution, analysis, & optimization
- Software goals
  - Universal interoperability (within & outside Trilinos)
  - Universal accessibility
    - Any hardware & operating system with a C++ compiler
    - Including programming languages besides C++
  - “Self-sustaining” software
    - Legible design & implementation
    - Sufficient testing & documentation for confident refactoring

Trilinos is made of packages

Trilinos is made of packages

- Not monolithic; ~60 separate packages
- Each package
  - Has its own development team & management
  - May or may not depend on other Trilinos packages
  - May even have a different license or release status
    - Most BSD; some LGPL
    - Some not publicly released yet (e.g., “pre-copyright”)
  - Benefits from Trilinos’ build, test, & release infrastructure
- Common build & test framework: TriBITS
  - Lets packages express their dependencies on
    - Other packages
    - Third-party libraries (e.g., HDF5, BLAS, SuperLU, …)
  - Runs packages’ tests nightly, & on every check-in
  - Useful: spun off from Trilinos into a separate project

Interoperability vs. Dependence

("Can Use") ("Depends On")

- Packages have minimal required dependencies…
- But interoperability makes them useful:
  - NOX (nonlinear solver) needs linear solvers
    - Can use any of {AztecOO, Belos, LAPACK, …}
  - Belos (linear solver) needs preconditioners, matrices, & vectors
    - Matrices and vectors: any of {Epetra, Tpetra, Thyra, …, PETSc}
    - Preconditioners: any of {IFPACK, ML, Ifpack2, MueLu, Teko, …}
- We express interoperability as "optional dependency"
- Interoperability is enabled at configure time
  - Each package declares its list of interoperable packages
  - Trilinos' build system automatically hooks them together
  - You tell Trilinos what packages you want to build…
  - …it automatically enables any packages on which it depends

Packages benefit users

- You decide how much of Trilinos you want
  - Only use & build the packages you need (not all ~60)
  - Can distribute any subset of packages
- Mix & match Trilinos components with your own, e.g.,
  - Trilinos' sparse matrices with your own linear solvers
  - Your sparse matrices with Trilinos' linear solvers
  - Trilinos' sparse matrices & linear solvers w/ your nonlinear solvers
- Easier to find specialized help for problems of your interest
  - Package developers know their packages' algorithms

Packages benefit developers

- Popular packages (e.g., ML & Zoltan) keep their “brand”
- Reflects organization of research / development teams
  - Easy to turn a research code into a new package
  - Small teams with minimal interference between teams
  - Teams have some freedom to define their own standards & schedule
- TriBITS build system supports external packages!
  - e.g., DataTransferKit: https://github.com/ORNL-CEES/DataTransferKit
  - Need not live in Trilinos’ repository or have Trilinos’ license

Capability areas and leaders

- Capability areas:
  - Framework, Tools, & Interfaces (Jim Willenbring)
  - Software Engineering Technologies & Integration (Ross Bartlett)
  - Discretizations (Mauro Perego)
  - Geometry, Meshing, & Load Balancing (Karen Devine)
  - Scalable Linear Algebra (Mark Hoemmen)
  - Linear & Eigen Solvers (Jonathan Hu)
  - Nonlinear, Transient, & Optimization Solvers (Andy Salinger)
  - Scalable I/O (Ron Oldfield)
  - User Experience (Bill Spotz)
- Each area includes one or more Trilinos packages
- Each leader provides strategic direction within area

Whirlwind Tour of Packages

Full Vertical Solver Coverage

Trilinos

Optimization **Unconstrained:** **Constrained:**	Find $u \in \Re^n$ that minimizes $g(u)$ Find $x \in \Re^m$ and $u \in \Re^n$ that minimizes $g(x,u)$ s.t. $f(x,u) = 0$	**Sensitivities (Automatic Differentiation: Sacado)**	**MOOCHO**
Bifurcation Analysis	Given nonlinear operator $F(x,u) \in \Re^{n+m}$ For $F(x,u) = 0$ find space $u \in U \ni \frac{\partial F}{\partial x}$		**LOCA**
Transient Problems **DAEs/ODEs:**	Solve $f(\dot{x}(t), x(t), t) = 0$ $t \in [0,T], x(0) = x_0, \dot{x}(0) = x_0'$ for $x(t) \in \Re^n, t \in [0,T]$		**Rythmos**
Nonlinear Problems	Given nonlinear operator $F(x) \in \Re^m \rightarrow \Re$ Solve $F(x) = 0$ $x \in \Re^n$		**NOX**
Linear Problems **Linear Equations:** **Eigen Problems:**	Given Linear Ops (Matrices) $A, B \in \Re^{m \times n}$ Solve $Ax = b$ for $x \in \Re^n$ Solve $A\nu = \lambda B\nu$ for (all) $\nu \in \Re^n$, $\lambda \in$		**AztecOO** **Belos** **Ifpack, ML, etc...** **Anasazi**
Distributed Linear Algebra **Matrix/Graph Equations** **Vector Problems:**	Compute $y = Ax; A = A(G); A \in \Re^{m \times n}, G \in \Im^{m \times n}$ Compute $y = \alpha x + \beta w; \alpha = \langle x, y \rangle; x, y \in \Re^n$		**Epetra** **Tpetra** **Kokkos**

Trilinos Package Summary

	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

Whirlwind Tour of Packages

Discretizations Methods Core Solvers

Trilinos' Common Language: Petra

- "Common language" for distributed sparse linear algebra

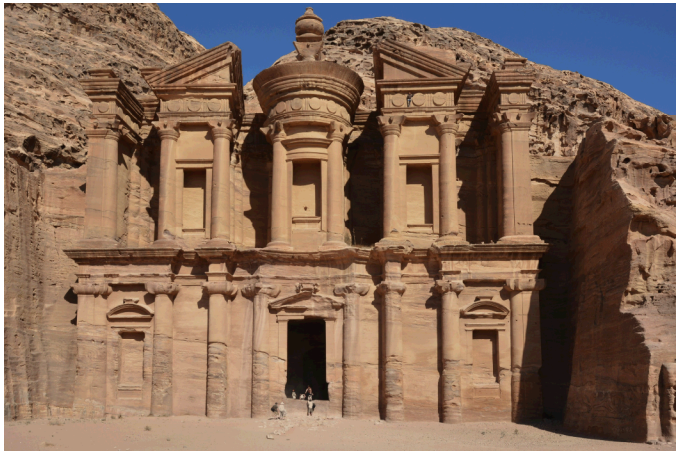
- Petra[1] provides parallel…
  - Sparse graphs & matrices
  - Dense vectors & multivectors
  - Data distributions & redistribution
- "Petra Object Model":
  - Describes objects & their relationships abstractly, independent of language or implementation
  - Explains how to construct, use, & redistribute parallel graphs, matrices, & vectors
- We maintain 2 implementations

Al Khazneh ("The Treasury"), in the ancient city of Petra, in modern Jordan.

[1]Petra (πέτρα) is Greek for "foundation."

Petra Implementations

- Epetra (Essential Petra):
  - Earliest & most heavily used
  - C++ <= 1998 (“C+/- compilers” OK)
  - Real, double-precision arithmetic
  - C & Fortran interfaces
  - MPI only (very little OpenMP support)
  - Some support for problems with over two billion unknowns (“Epetra64”)
- Tpetra (Templated Petra):
  - Supports & requires C++11 (as of 11.14)
  - Real, complex, extended-precision, automatic differentiation, etc. types
  - Can solve problems with > 2B unknowns
  - “MPI+X” (shared-memory parallel)



Al Deir (“The Monastery”) at Petra.

Package leads: Mike Heroux, Mark Hoemmen (many developers & contributors)

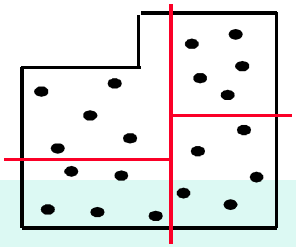
Two "software stacks": Epetra & Tpetra

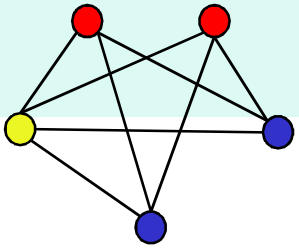
- Many packages were built on Epetra's interface
- Users want features that break interfaces
  - Support for solving huge problems (> 2B entities)
  - Arbitrary & mixed precision
  - Hybrid (MPI+X) parallelism (← most radical interface changes)
- Users also value backwards compatibility
- We decided to build a (partly) new stack using Tpetra
- Some packages can work with either Epetra or Tpetra
  - Iterative linear solvers & eigensolvers (Belos, Anasazi)
  - Multilevel preconditioners (MueLu), sparse direct (Amesos2)
- Which do I use?
  - Epetra is more stable; Tpetra is more forward-looking
  - For MPI only, their performance is comparable
  - For MPI+X, Tpetra will be the only path forward

Kokkos: Thread-parallel programming model & more

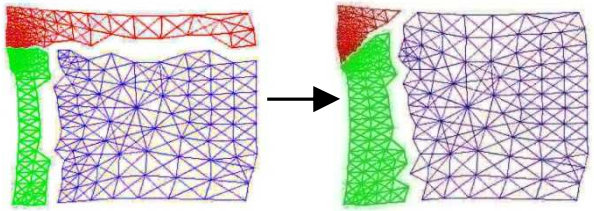
- Performance-portable abstraction over many different thread-parallel programming models: OpenMP, CUDA, Pthreads, …
  - Avoid risk of committing code to hardware or programming model
  - C++ library: Widely used, portable language with good compilers
- Abstract away physical data layout & target it to the hardware
  - Solve “array of structs” vs. “struct of arrays” problem
- Expose different memory & execution spaces
- Data structures & idioms for thread-scalable parallel code
  - Multi-dimensional arrays, hash table, sparse graph & matrix
  - Automatic memory management, atomic updates, vectorization, ...
- Stand-alone; does not require other Trilinos packages
  - Used in LAMMPS molecular dynamics code
  - Growing use in Trilinos; other apps starting too
- Talk this week about task parallelism: Tue 11:15

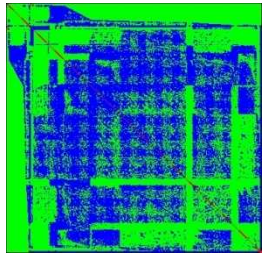
Zoltan(2)



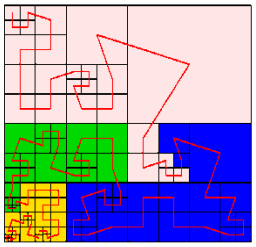


- Data Services for Dynamic Applications
  - Dynamic load balancing
  - Graph coloring
  - Data migration
  - Matrix ordering

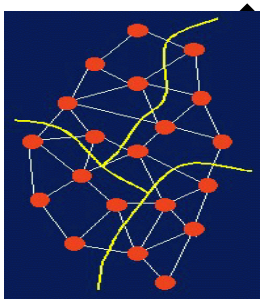




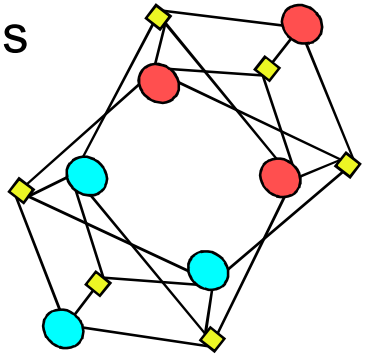
- Partitioners:



Geometric (coordinate-based) methods:
- Recursive Coordinate Bisection
- Recursive Inertial Bisection
- Space Filling Curves
- Refinement-tree Partitioning



Hypergraph & graph (connectivity-based) methods



Isorropia package: interface to Epetra objects

Zoltan2: interface to Tpetra objects

Developers: Karen Devine, Erik Boman, Siva Rajamanickam, Michael Wolf

Sandia National Laboratories

Python & web "skins"

- PyTrilinos (Python interface to Trilinos)
  - Wrappers for many Trilinos packages
  - Uses SWIG to generate bindings
  - See talk Wed 8:30

Lead: Bill Spotz

- WebTrilinos (web interface to Trilinos)
  - Generate test problems or read from file
  - Edit C++ / Python code fragments; click to build & run

Lead: Jim Willenbring

- MueMex (Matlab interface to MueLu)
  - … & thereby to many Trilinos packages
  - See talk Tue 15:30

(Maintained by MueLu developers)

Whirlwind Tour of Packages

Discretizations Methods Core Solvers

AztecOO

- Iterative linear solvers: CG, GMRES, BiCGSTAB,…
- Incomplete factorization preconditioners

- Aztec was Sandia's workhorse solver:
  - Extracted from the MPSalsa reacting flow code
  - Installed in dozens of Sandia apps
  - 1900+ external licenses
- AztecOO improves on Aztec by:
  - Using Epetra objects for defining matrix and vectors
  - Providing more preconditioners & scalings
  - Using C++ class design to enable more sophisticated use
- AztecOO interface allows:
  - Continued use of Aztec for functionality
  - Introduction of new solver capabilities outside of Aztec

Developers: Mike Heroux, Alan Williams, Ray Tuminaro

Belos

- Next-generation linear iterative solvers

- Decouples algorithms from linear algebra objects
  - Supports any linear algebra library (Epetra, Tpetra, Thyra, …)
  - Linear algebra library has full control over data layout and kernels
  - Improvement over AztecOO, which controlled vector & matrix layout
  - Essential for hybrid (MPI+X) parallelism
- Solves problems that apps really want to solve, faster:
  - Multiple right-hand sides: AX=B
  - Sequences of related systems: $(A + \Delta A_k) X_k = B + \Delta B_k$
- Many advanced methods for these types of systems
  - Block & pseudoblock solvers: GMRES & CG
  - Recycling solvers: GCRODR (GMRES) & CG
  - “Seed” solvers (hybrid GMRES)
  - Block orthogonalizations (TSQR)
- Supports arbitrary & mixed precision, complex, …

Developers: Heidi Thornquist, Mike Heroux, Alicia Klinvex, Mark Hoemmen

Amesos(2)

- Interface to sparse direct solvers
- Challenges:
  - Many different third-party sparse direct solvers
  - No clear winner for all problems
  - Different, changing interfaces & data formats, serial & parallel
- Amesos(2) features:
  - Accepts Epetra & Tpetra sparse matrices & dense vectors
  - Consistent interface to many third-party sparse factorizations
    - e.g., MUMPS, Paradiso, SuperLU, Taucs, UMFPACK
  - Can use factorizations as solvers in other Trilinos packages
  - Automatic efficient data redistribution (if solver not MPI parallel)
  - Built-in default solvers: KLU(2), Basker (Amesos2 only)
  - Interface to new direct / iterative hybrid-parallel solver ShyLU

Developers: Ken Stanley, Marzio Sala, Tim Davis; Eric Bavier, Erik Boman, Siva Rajamanickam

Ifpack(2): Algebraic preconditioners

- Preconditioners:
  - Overlapping domain decomposition
  - Incomplete factorizations (within an MPI process)
  - (Block) relaxations & Chebyshev
- Accepts user matrix via abstract matrix interface
- Use {E,T}petra for basic matrix / vector calculations
- Perturbation stabilizations & condition estimation
- Can be used by all other Trilinos solver packages
- Ifpack2: Tpetra version of Ifpack
  - Supports arbitrary precision & complex arithmetic
  - Path forward to hybrid-parallel factorizations

: Multi-level Preconditioners

- Smoothed aggregation, multigrid, & domain decomposition

- Critical technology for scalable performance of many apps
- ML compatible with other Trilinos packages:
  - Accepts Epetra sparse matrices & dense vectors
  - ML preconditioners can be used by AztecOO, Belos, & Anasazi
- Can also be used independent of other Trilinos packages
- Next-generation version of ML: MueLu
  - Works with Epetra or Tpetra objects (via Xpetra interface)

Developers: Ray Tuminaro, Jeremie Gaidamour, Jonathan Hu, Marzio Sala, Chris Siefert

MueLu: Next-gen algebraic multigrid

- Motivation for replacing ML
  - Improve maintainability & ease development of new algorithms
  - Decouple computational kernels from algorithms
  - Rely more on other Trilinos packages for linear algebra, smoothers, etc.
- Exploit Tpetra features
  - MPI+X (Kokkos programming model mitigates risk)
  - 64-bit global indices (to solve problems with >2B unknowns)
  - Arbitrary Scalar types (Tramonto runs MueLu w/ double-double)
- Works with Epetra or Tpetra (via Xpetra common interface)
- Facilitate algorithm development
  - Energy minimization methods
  - Geometric or classic algebraic multigrid; mix methods together
- Better preconditioner reuse (for nonlinear solve iterations)
- Amazing new tutorial and users' guide (Tobias Wiesner)!

Developers: Andrey Prokopenko, Jonathan Hu, Chris Siefert, Ray Tuminaro, Tobias Wiesner

Anasazi

- Next-generation iterative eigensolvers

- Decouples algorithms from linear algebra objects
- Block eigensolvers for accurate cluster resolution
- Can solve
  - Standard ($AX = \Lambda X$) or generalized ($AX = BX\Lambda$)
  - Hermitian or not, real or complex
- Algorithms available
  - Block Krylov-Schur (most like ARPACK's IR Arnoldi) & Block Davidson
  - Implicit Riemannian Trust Region solvers
  - Locally Optimal Block-Preconditioned CG (LOBPCG)
  - Scalable orthogonalizations (e.g., TSQR, SVQB)

Developers: Heidi Thornquist, Mike Heroux, Chris Baker, Rich Lehoucq, Ulrich Hetmaniuk, Alicia Klinvex, Mark Hoemmen

NOX: Nonlinear Solvers

- Suite of nonlinear solution methods

Broyden's Method

$$M_B = f(x_c) + B_c d$$

Newton's Method

$$M_N = f(x_c) + J_c d$$

Tensor Method

$$M_T = f(x_c) + J_c d + \frac{1}{2} T_c d d$$

Jacobian Estimation

- Graph Coloring
- Finite Difference
- Jacobian-Free Newton-Krylov

Globalizations

Line Search	Trust Region
Interval Halving	Dogleg
Quadratic	Inexact Dogleg
Cubic	
More-Thuente	

Implementation

- Parallel
- Object-oriented C++
- Independent of the linear algebra implementation!

Developers: Tammy Kolda, Roger Pawlowski

Sandia National Laboratories

LOCA: Continuation problems

- Solve parameterized nonlinear systems $F(x, \lambda) = 0$
- Identify “interesting” nonlinear behavior, like
  - Turning points
    - Buckling of a shallow arch under symmetric load
    - Breaking away of a drop from a tube
    - Bursting of a balloon at a critical volume
  - Bifurcations, e.g., Hopf or pitchfork
    - Vortex shedding (e.g., turbulence near mountains)
    - Flutter in airplane wings; electrical circuit oscillations
- LOCA uses Trilinos components
  - NOX to solve nonlinear systems
  - Anasazi to solve eigenvalue problems
  - AztecOO or Belos to solve linear systems

Developers: Andy Salinger, Eric Phipps

- Rapid Optimization Library (C++)
- Optimal {design, control} & inverse problems
- Other uses: mesh optimization, image processing
- Consolidates 4 of Trilinos' existing packages:
  - Aristos, MOOCHO, OptiPack, GlobiPack
- Key features:
  - Matrix free: Direct use of app's data structures & solvers
  - State-of-the-art algorithms
    - For (un)constrained optimization & opt. under uncertainty
    - Enable {inexact, adaptive} function evaluations & linear solves
  - Special APIs for simulation-based optimization
    - Streamline embedding into engineering apps
  - Modular interfaces: enable custom & user-defined algorithms, stopping criteria, hierarchies of algorithms, etc.

Developers: Drew Kouri, Denis Ridzal, Bart van Bloemen Waanders, Greg von Winckel

Sandia National Laboratories

Whirlwind Tour of Packages

Core Utilities

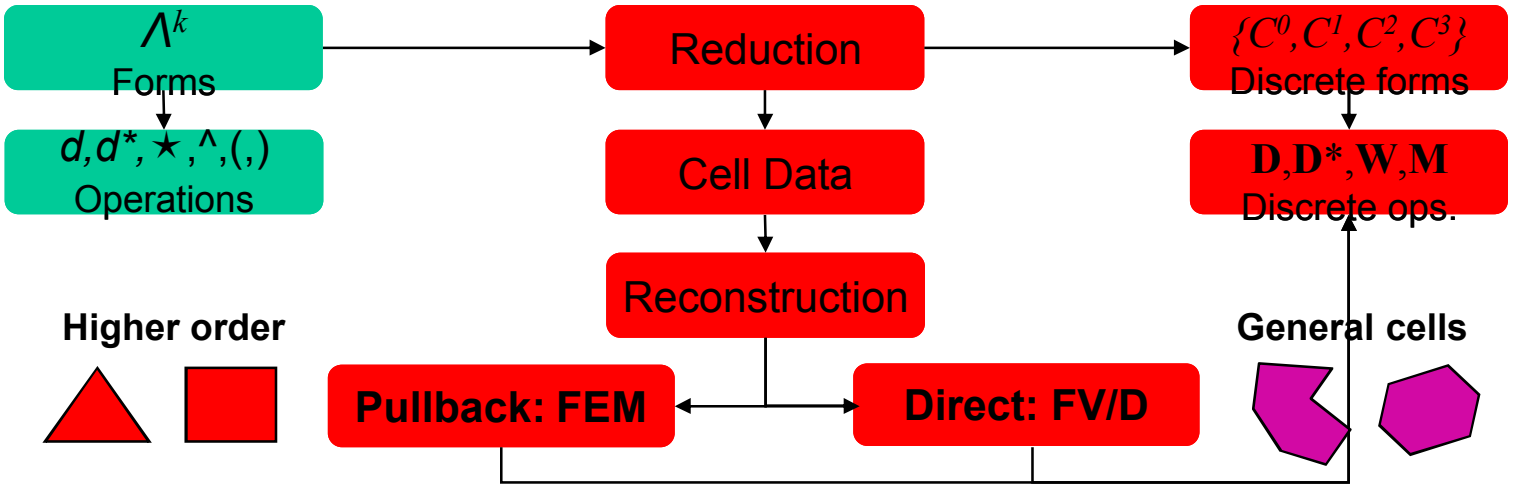
Discretizations Methods Solvers

Intrepid

Interoperable Tools for Rapid Development of Compatible Discretizations

Intrepid offers an **innovative software design** for compatible discretizations:

- Access to finite {element, volume, difference} methods using a common API
- Supports hybrid discretizations (FEM, FV and FD) on unstructured grids
- Supports a variety of cell shapes:
  - Standard shapes (e.g., tets, hexes): high-order finite element methods
  - Arbitrary (polyhedral) shapes: low-order mimetic finite difference methods
- Enables optimization, error estimation, V&V, & UQ using fast embedded techniques (direct support for cell-based derivative computations or via automatic differentiation)



Developers: Pavel Bochev & Denis Ridzal

See talk Wed 10:20

Rythmos

- Numerical time integration methods
- “Time integration” == “ODE solver”
- Supported methods include
  - Backward & Forward Euler
  - Explicit Runge-Kutta
  - Implicit BDF
- Operator splitting methods & sensitivities

Developers: Glen Hansen, Roscoe Bartlett, Todd Coffey

Whirlwind Tour of Packages

Discretizations Methods Core Solvers

Sacado: Automatic Differentiation

- Automatic differentiation tools optimized for element-level computation

- Applications of AD: Jacobians, sensitivity and uncertainty analysis, …
- Uses C++ templates to compute derivatives
  - You maintain one templated code base; derivatives don't appear explicitly
- Provides three forms of AD
  - Forward Mode: $(x,\ V) \longrightarrow \left(f,\ \frac{\partial f}{\partial x} V\right)$
    - Propagate derivatives of intermediate variables w.r.t. independent variables forward
    - Directional derivatives, tangent vectors, square Jacobians, ∂f / ∂x when **m ≥ n**
  - Reverse Mode: $(x,\ W) \longrightarrow \left(f,\ W^T \frac{\partial f}{\partial x}\right)$
    - Propagate derivatives of dependent variables w.r.t. intermediate variables backwards
    - Gradients, Jacobian-transpose products (adjoints), ∂f / ∂x when **n > m**.
  - Taylor polynomial mode: $x(t) = \sum_{k=0}^{d} x_k t^k \longrightarrow \sum_{k=0}^{d} f_k t^k = f(x(t)) + O(t^{d+1}),\quad f_k = \frac{1}{k!}\frac{d^k}{dt^k} f(x(t))$
  - Basic modes combined for higher derivatives

Developers: Eric Phipps, David Gay

Sandia National Laboratories

Questions?