An Overview of Trilinos

Mark Hoemmen
Sandia National Laboratories
18 August 2011

SAND 2011-5957C

Sandia National Laboratories

Who am I?

- Postdoc at Sandia National Laboratories
  - Albuquerque, New Mexico, USA
- Research: “Scalable algorithms”
  - New Krylov subspace methods ($Ax=b$, $Ax=\lambda x$)
  - Parallel programming models
- Trilinos developer since Spring 2010
  - New, fast, accurate block orthogonalization (TSQR)
  - New iterative linear solvers in progress
  - Sparse matrix I/O, utilities, bug fixes, and consulting
- Trilinos packages I’ve worked on:
  - Anasazi, Belos, Kokkos, Teuchos, Tpetra
- I can’t answer every question, but will try my best!

Outline

- What can Trilinos do for you?
- Trilinos' software organization
- Whirlwind tour of Trilinos packages
- Getting started: "How do I…?"
- Preparation for hands-on tutorial

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™

Sandia National Laboratories

Trilinos Contributors

Chris Baker
Ross Bartlett
Pavel Bochev
Paul Boggs
Erik Boman
Lee Buermann
Cedric Chevalier
Todd Coffey
Eric Cyr
David Day
Karen Devine
Clark Dohrmann
Kelly Fermoyle
David Gay
Jeremie Gaidamour
Mike Heroux
Ulrich Hetmaniuk
Mark Hoemmen
Russell Hooper
Vicki Howle

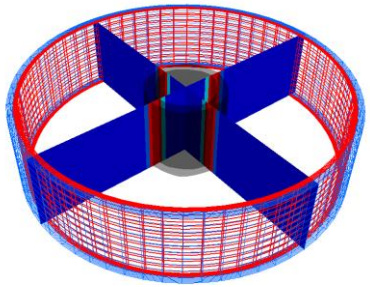
Jonathan Hu
Joe Kotulski
Rich Lehoucq
Kevin Long
Karla Morris
Kurtis Nusbaum
Roger Pawlowski
Brent Perschbacher
Eric Phipps
Siva Rajamanickam
Lee Ann Riesen
Marzio Sala
Andrew Salinger
Nico Schlömer
Chris Siefert
Bill Spotz
Heidi Thornquist
Ray Tuminaro
Jim Willenbring
Alan Williams
Michael Wolf

Past Contributors

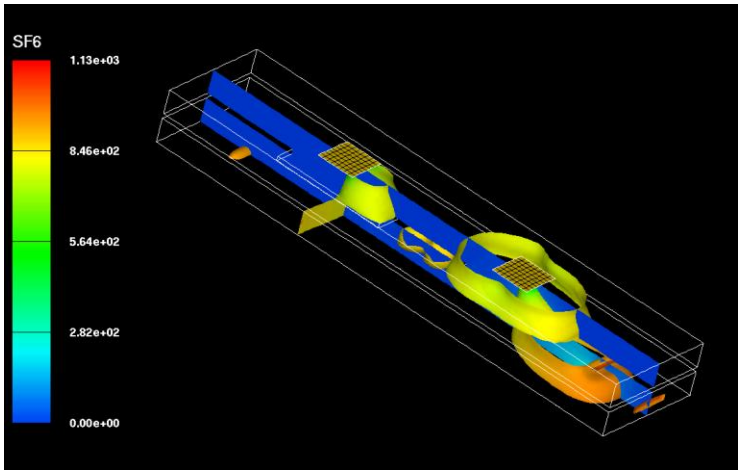
Jason Cross
Michael Gee
Esteban Guillen
Bob Heaphy
Robert Hoekstra
Kris Kampshoff
Ian Karlin
Sarah Knepper
Tammy Kolda
Joe Outzen
Mike Phenow
Paul Sexton
Bob Shuttleworth
Ken Stanley

Sandia National Laboratories

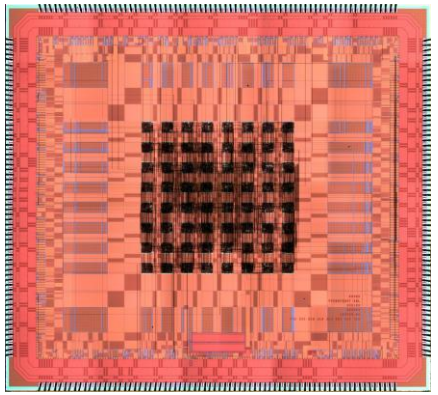
Applications

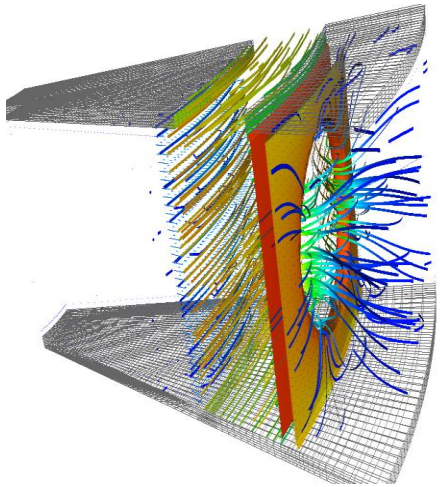


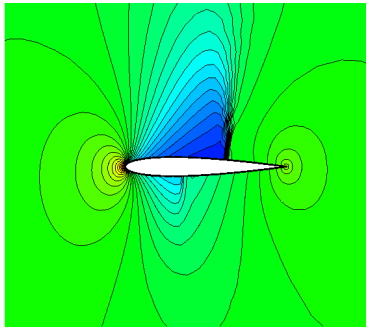
PDEs



Circuits

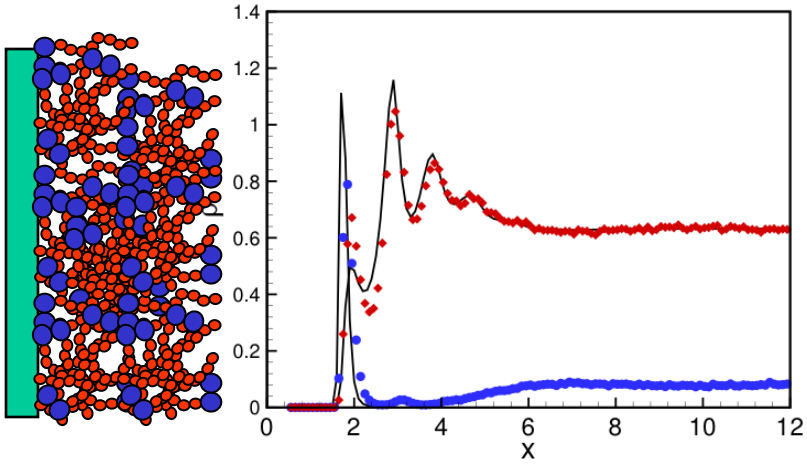






And More…

Inhomogeneous Fluids



Applications

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

Target platforms:
Any and all, current and future

- Laptops and workstations
- Clusters and supercomputers
  - Multicore CPU nodes
  - Hybrid CPU / GPU nodes
- Parallel programming environments
  - MPI, OpenMP
  - Intel TBB, Pthreads
  - Thrust, CUDA
  - Combinations of the above
- User “skins”
  - C++ (primary language)
  - C, Fortran, Python
  - Web (today’s hands-on!)

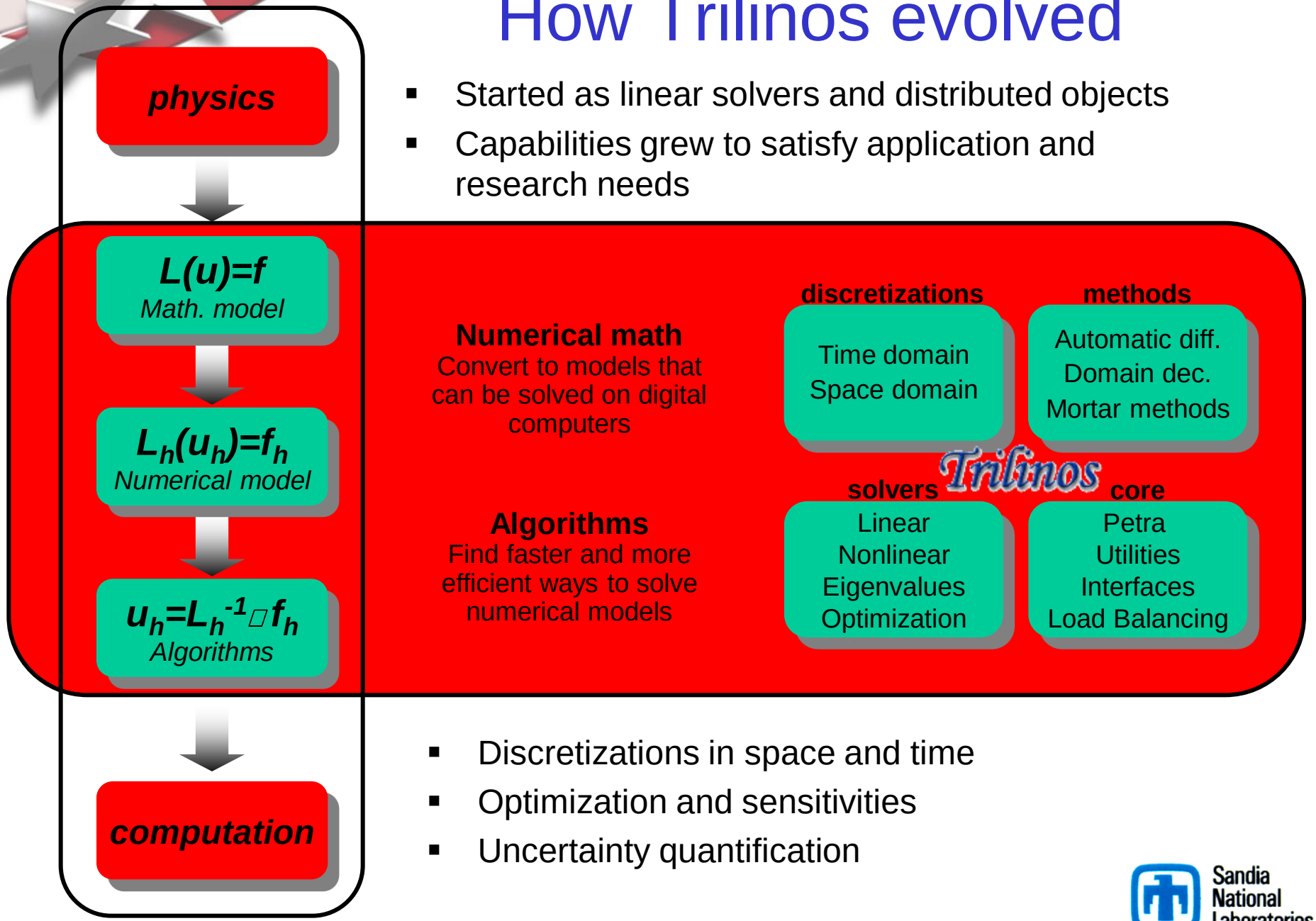
Sandia National Laboratories

Unique features of Trilinos

- Huge library of algorithms
  - Linear and nonlinear solvers, preconditioners, …
  - Optimization, transients, sensitivities, uncertainty, …
- Growing support for multicore & hybrid CPU/GPU
  - Built into the new Tpetra linear algebra objects
    - Therefore into iterative solvers with zero effort!
  - Unified intranode programming model
  - Spreading into the whole stack:
    - Multigrid, sparse factorizations, element assembly…
- Growing support for mixed and arbitrary precisions
  - Don’t have to rebuild Trilinos to use it!
- Growing support for huge (> 2B unknowns) problems

How Trilinos evolved

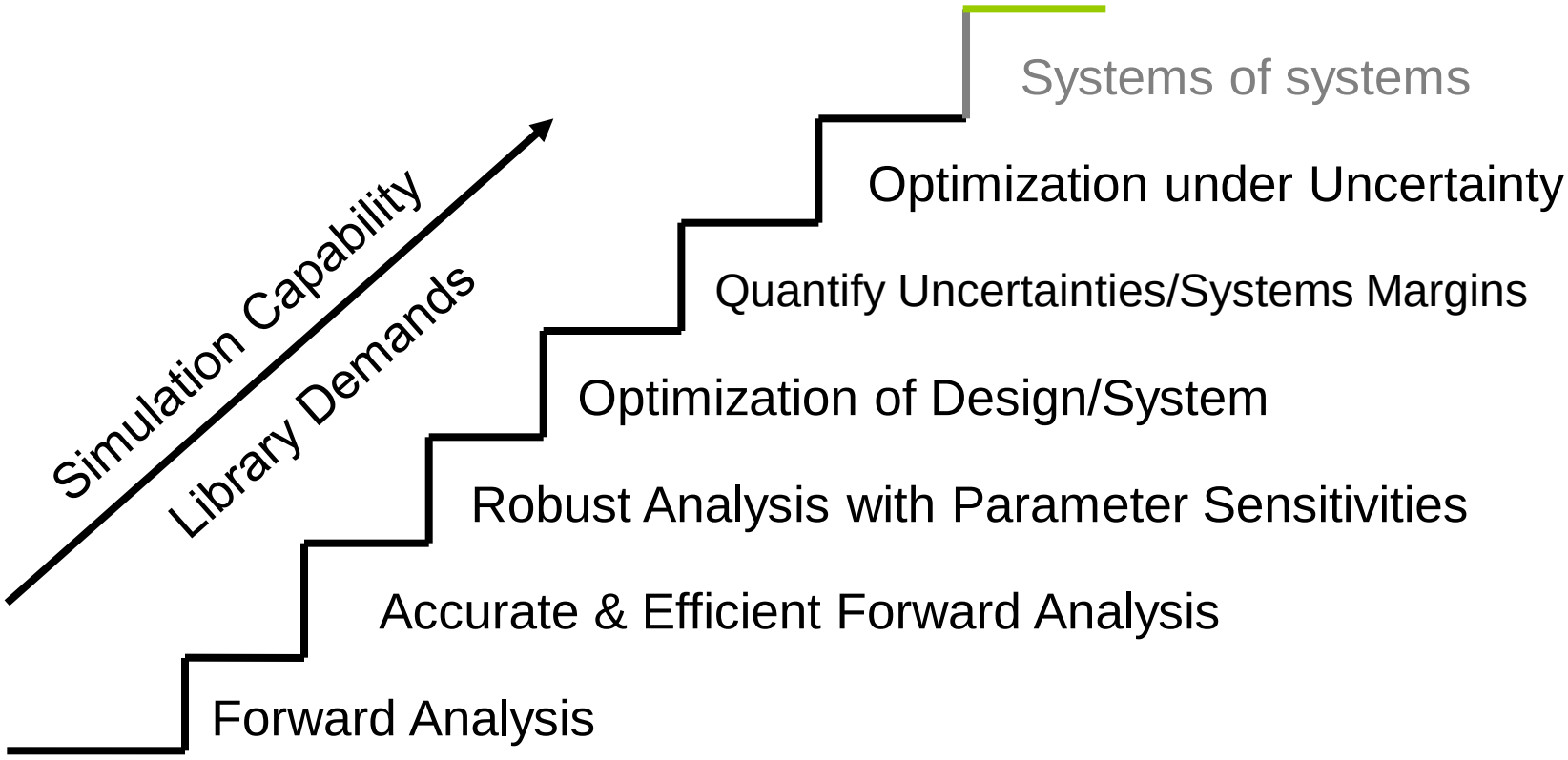
- Started as linear solvers and distributed objects
- Capabilities grew to satisfy application and research needs



- Discretizations in space and time
- Optimization and 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
  - Hardened computations
    - Fail only if problem intractable
    - Diagnose failures and inform the user
  - Full vertical coverage
    - Problem construction, solution, analysis, and optimization
- Software goals
  - Universal interoperability
    - Between any Trilinos components
    - With external software (PETSc, Hypre, …)
  - Universal accessibility
    - Programming languages, hardware, operating systems

Trilinos' software organization

Trilinos is made of packages

- Not a monolithic piece of software
  - Like LEGO™ bricks, not Matlab™
- Each package:
  - Has its own development team and management
  - Makes its own decisions about algorithms, coding style, etc.
  - May or may not depend on other Trilinos packages
- Trilinos is not “indivisible”
  - You don’t need all of Trilinos to get things done
  - Any subset of packages can be combined and distributed
  - Current public release contains ~50 of the 55+ Trilinos packages
- Trilinos top layer framework
  - Not a large amount of source code: ~1.5%
  - Manages package dependencies
    - Like a GNU/Linux package manager
  - Runs packages’ tests nightly, and on every check-in
- Package model supports multifrontal development

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, and vectors
    - Matrices and vectors: any of {Epetra, Tpetra, Thyra, …, PETSc}
    - Preconditioners: any of {IFPACK, ML, Teko, …}
- Interoperability is enabled at configure time
  - Each package declares its list of interoperable packages
  - Trilinos' CMake system automatically hooks them together

Capability areas and leaders

- Capability areas:
  - Framework, Tools & Interfaces (Jim Willenbring)
  - Software Engineering Technologies and Integration (Ross Bartlett)
  - Discretizations (Pavel Bochev)
  - Geometry, Meshing & Load Balancing (Karen Devine)
  - Scalable Linear Algebra (Mike Heroux)
  - Linear & Eigen Solvers (Jonathan Hu)
  - Nonlinear, Transient & Optimization Solvers (Andy Salinger)
  - Scalable I/O: (Ron Oldfield)
- 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	STKMesh, Intrepid, Pamgen, Sundance, ITAPS, Mesquite
	Time Integration	Rythmos
Methods	Automatic Differentiation	Sacado
	Mortar Methods	Moertel
Services	Linear algebra objects	Epetra, Tpetra, Kokkos
	Interfaces	Thyra, Stratimikos, RTOp, FEI, Shards, Tpetra::RTI
	Load Balancing	Zoltan, Isorropia
	“Skins”	PyTrilinos, WebTrilinos, ForTrilinos, Ctrilinos, Optika
	C++ utilities, I/O, thread API	Teuchos, EpetraExt, Kokkos, Triutils, ThreadPool, Phalanx
Solvers	Iterative linear solvers	AztecOO, Belos, Komplex
	Direct sparse linear solvers	Amesos, Amesos2
	Direct dense linear solvers	Epetra, Teuchos, Pliris
	Iterative eigenvalue solvers	Anasazi
	ILU-type preconditioners	AztecOO, IFPACK, Ifpack2
	Multilevel preconditioners	ML, CLAPS
	Block preconditioners	Meros, Teko
	Nonlinear system solvers	NOX, LOCA
	Optimization (SAND)	MOOCHO, Aristos, TriKota, Globipack, Optipack
	Stochastic PDEs	Stokhos

Whirlwind Tour of Packages

Core Utilities

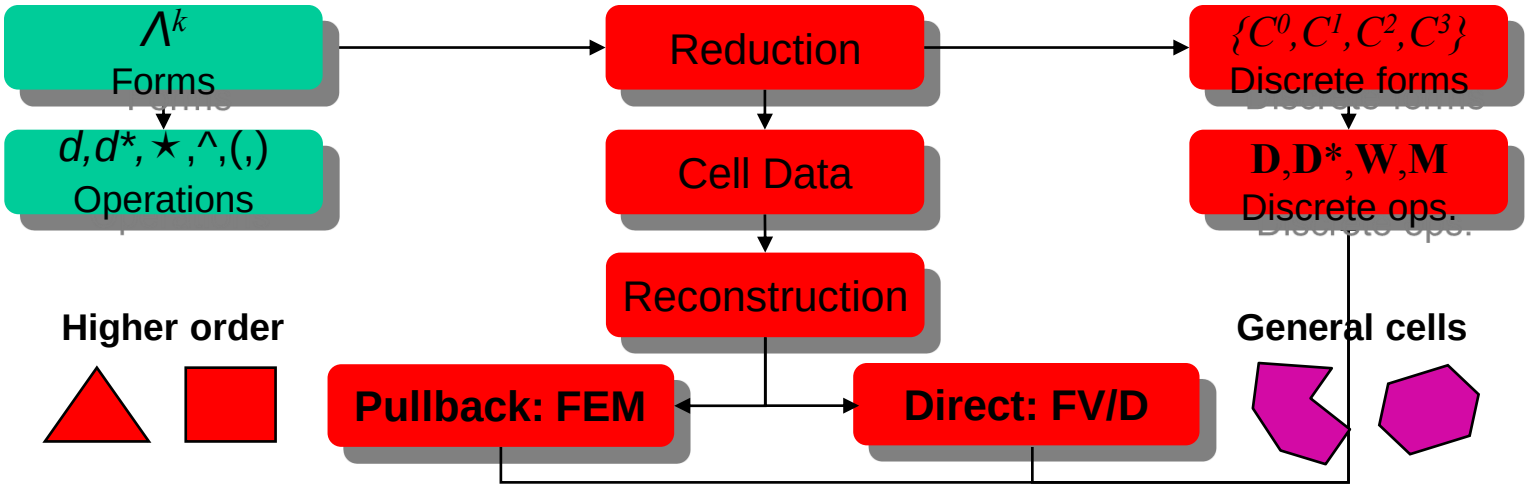
Discretizations **Methods** **Solvers**

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, and UQ using fast invasive techniques (direct support for cell-based derivative computations or via automatic differentiation)



Developers: Pavel Bochev and Denis Ridzal

Rythmos

- Suite of time integration (discretization) methods

- Currently includes:
  - Backward and Forward Euler
  - Explicit Runge-Kutta
  - Implicit BDF at this time.
- Native support for operator splitting methods
- Highly modular
- Forward sensitivities included in first release
- Adjoint sensitivities coming soon

Developers: Todd Coffey, Roscoe Bartlett

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

Whirlwind Tour of Packages

Discretizations **Methods** **Core** **Solvers**

Teuchos

- Portable utility package of commonly useful tools
  - ParameterList: nested key-value pair database (more later)
  - LAPACK, BLAS wrappers (templated on ordinal and scalar type)
  - Dense matrix and vector classes (compatible with BLAS/LAPACK)
  - Memory management classes (more later)
  - Scalable parallel timers and statistics
  - Support for generic algorithms (traits classes)
- Takes advantage of advanced features of C++:
  - Templates
  - Standard Template Library (STL)

Developers: Chris Baker, Roscoe Barlett, Mike Heroux, Mark Hoemmen, Kris Kampshoff, Kevin Long, Paul Sexton, Heidi Thornquist

Sandia National Laboratories

Trilinos Common Language: Petra

- Petra provides a “common language” for distributed linear algebra objects (operator, matrix, vector)

- Petra[1] provides distributed matrix and vector services
- Exists in basic form as an object model:
  - Describes basic user and support classes in UML, independent of language/implementation
  - Describes objects and relationships to build and use matrices, vectors and graphs
- Has 2 implementations under active development

[1]Petra is Greek for “foundation”.

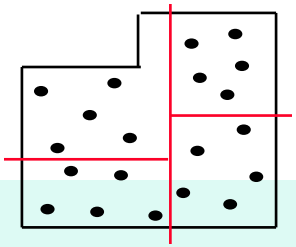
Petra Implementations

- Epetra (Essential Petra):
  - Current production version
  - Uses stable core subset of C++ (circa 2000)
  - Restricted to real, double-precision arithmetic
  - Interfaces accessible to C and Fortran users

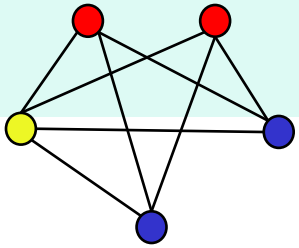
- Tpetra (Templated Petra):
  - Next-generation version
  - Needs a modern C++ compiler (but not C++0x)
  - Supports arbitrary scalar and index types via templates
    - Arbitrary- and mixed-precision arithmetic
    - 64-bit indices for solving problems with >2 billion unknowns
  - Hybrid MPI / shared-memory parallel
    - Supports multicore CPU and hybrid CPU/GPU
    - Built on Kokkos manycore node library

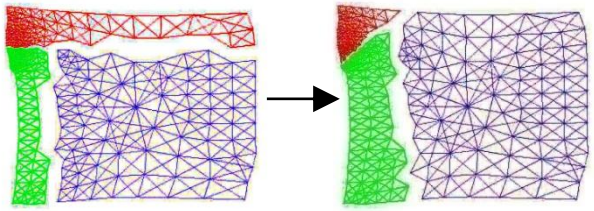
Developers: Chris Baker, Mike Heroux, Rob Hoekstra, Alan Williams

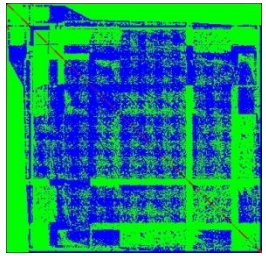
Zoltan



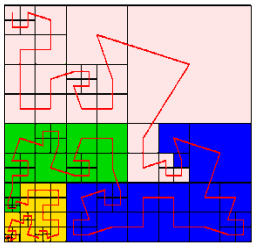
- Data Services for Dynamic Applications
  - Dynamic load balancing
  - Graph coloring
  - Data migration
  - Matrix ordering



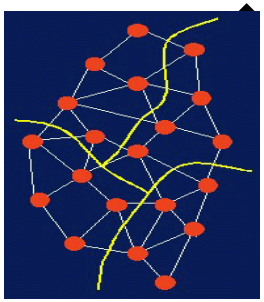




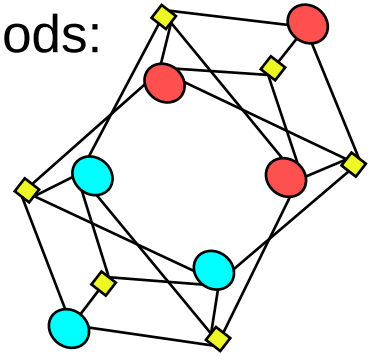
- Partitioners:



Geometric (coordinate-based) methods:
- Recursive Coordinate Bisection (Berger, Bokhari)
- Recursive Inertial Bisection (Taylor, Nour-Omid)
- Space Filling Curves (Peano, Hilbert)
- Refinement-tree Partitioning (Mitchell)



Hypergraph and graph (connectivity-based) methods:
- Hypergraph Repartitioning PaToH (Catalyurek)
- Zoltan Hypergraph Partitioning
- ParMETIS (U. Minnesota)
- Jostle (U. Greenwich)



Isorropia package: interface to Epetra objects

Developers: Karen Devine, Eric Boman, Robert Heaphy, Siva Rajamanickam

Thyra

- High-performance, abstract interfaces for linear algebra
- Offers flexibility through abstractions to algorithm developers
- Linear solvers (Direct, Iterative, Preconditioners)
  - Abstraction of basic vector/matrix operations (dot, axpy, mv).
  - Can use any concrete linear algebra library (Epetra, PETSc, BLAS).
- Nonlinear solvers (Newton, etc.)
  - Abstraction of linear solve (solve $Ax=b$).
  - Can use any concrete linear solver library:
    - AztecOO, Belos, ML, PETSc, LAPACK
- Transient/DAE solvers (implicit)
  - Abstraction of nonlinear solve.
  - … and so on.

Developers: Roscoe Bartlett, Kevin Long

“Skins”

- PyTrilinos provides Python access to Trilinos packages
- Uses SWIG to generate bindings.
- Epetra, AztecOO, IFPACK, ML, NOX, LOCA, Amesos and NewPackage are supported.

Developer: Bill Spotz

- CTrilinos: C wrapper (mostly to support ForTrilinos).
- ForTrilinos: OO Fortran interfaces.

Developers: Nicole Lemaster, Damian Rouson

- WebTrilinos: Web interface to Trilinos
- Generate test problems or read from file.
- Generate C++ or Python code fragments and click-run.
- Hand modify code fragments and re-run.
- Will use during hands-on.

Developers: Ray Tuminaro, Jonathan Hu, Marzio Sala, Jim Willenbring

Whirlwind Tour of Packages

Discretizations **Methods** **Core** **Solvers**

Sandia National Laboratories

Amesos

- Interface to direct solvers for distributed sparse linear systems (KLU, UMFPACK, SuperLU, MUMPS, ScaLAPACK)

- Challenges:
  - No single solver dominates
  - Different interfaces and data formats, serial and parallel
  - Interface often changes between revisions
- Amesos offers:
  - A single, clear, consistent interface, to various packages
  - Common look-and-feel for all classes
  - Separation from specific solver details
  - Use serial and distributed solvers; Amesos takes care of data redistribution
  - Native solvers: KLU and Paraklete

Developers: Ken Stanley, Marzio Sala, Tim Davis

Amesos2

- Second-generation sparse direct solvers package

- Unified interface to multiple solvers, just like Amesos
- Amesos2 features:
  - Supports matrices of arbitrary scalar and index types
  - Path to multicore CPU and hybrid CPU/GPU solvers
  - Thread safe: multiple solvers can coexist on the same node
    - Supports new intranode hybrid direct / iterative solver ShyLU
  - Abstraction from specific sparse matrix representation
    - Supports Epetra and Tpetra
    - Extensible to other matrix types
- September 2011 release

Developers: Eric Bavier, Erik Boman, and Siva Rajamanickam

AztecOO

- Krylov subspace solvers: CG, GMRES, BiCGSTAB,…
- Incomplete factorization preconditioners

- Aztec is 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
  - Better than “reverse communication” interface of Aztec
  - Linear algebra library controls storage and kernels
  - Essential for multicore CPU / GPU nodes
- 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 methods: Block GMRES and Block CG
  - Recycling solvers: GCRODR (GMRES) and CG
  - “Seed” solvers (hybrid GMRES)
  - Block orthogonalizations (TSQR)
- Supports arbitrary and mixed precision, and complex

Developers: Heidi Thornquist, Mike Heroux, Mark Hoemmen, Mike Parks, Rich Lehoucq

Sandia National Laboratories

IFPACK: Algebraic Preconditioners

- Overlapping Schwarz preconditioners with incomplete factorizations, block relaxations, & block direct solves.
- Accepts user matrix via abstract matrix interface
- Uses Epetra for basic matrix/vector calculations
- Simple perturbation stabilizations and condition est.
- Can be used by NOX, ML, AztecOO, Belos, …

Developers: Marzio Sala, Mike Heroux, Siva Rajamanickam, Alan Williams

Ifpack2

- Second-generation IFPACK

- Highly optimized ILUT (60x faster than IFPACK's!)
- Computed factors fully exploit multicore CPU / GPU
  - Via Tpetra
- Path to hybrid-parallel factorizations
- Arbitrary precision and complex arithmetic support

Developers: Mike Heroux, Siva Rajamanickam, Alan Williams, Michael Wolf

: Multi-level Preconditioners

- Smoothed aggregation, multigrid and domain decomposition preconditioning package

- Critical technology for scalable performance of many apps
- ML compatible with other Trilinos packages:
  - Accepts user data as Epetra_RowMatrix object (abstract interface). Any implementation of Epetra_RowMatrix works.
  - Implements the Epetra_Operator interface. Allows ML preconditioners to be used with AztecOO, Belos, Anasazi.
- Can also be used independent of other Trilinos packages

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

Anasazi

- Next-generation iterative eigensolvers

- Decouples algorithms from linear algebra objects
  - Better than “reverse communication” interface of ARPACK
  - Linear algebra library controls storage and kernels
  - Essential for multicore CPU / GPU nodes
- 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
  - Locally Optimal Block-Preconditioned CG (LOBPCG)
  - Implicit Riemannian Trust Region solvers
  - Advanced (faster & more accurate) orthogonalizations

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

Sandia National Laboratories

NOX: Nonlinear Solvers

- Suite of nonlinear solution methods

Broyden's Method

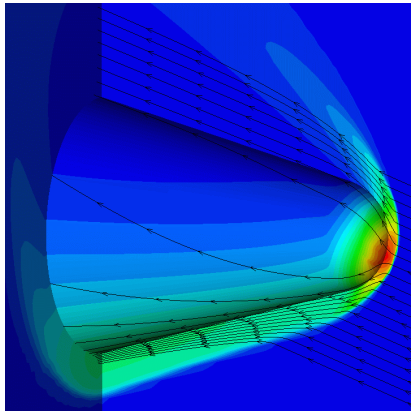
$$M_B = f(x_c) + B_c d$$

Newton's Method

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

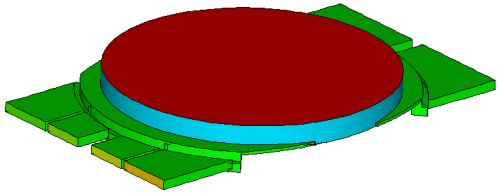
Tensor Method

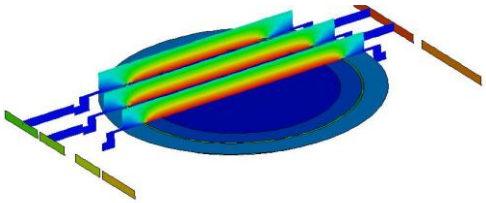
$$M_T = f(x_c) + J_c d + \frac{1}{2} T_c dd$$

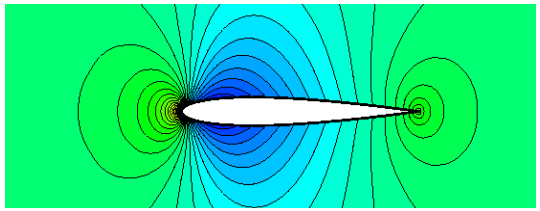


Globalizations

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







Jacobian Estimation

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

http://trilinos.sandia.gov/packages/nox

Implementation

- Parallel
- OO-C++
- Independent of the linear algebra package!

Developers: Tammy Kolda, Roger Pawlowski

National Laboratories

LOCA

- Library of continuation algorithms

- Provides
  - Zero order continuation
  - First order continuation
  - Arc length continuation
  - Multi-parameter continuation (via Henderson's MF Library)
  - Turning point continuation
  - Pitchfork bifurcation continuation
  - Hopf bifurcation continuation
  - Phase transition continuation
  - Eigenvalue approximation (via ARPACK or Anasazi)

Developers: Andy Salinger, Eric Phipps

MOOCHO & Aristos

- MOOCHO: Multifunctional Object-Oriented arCHitecture for Optimization
  - Large-scale invasive simultaneous analysis and design (SAND) using reduced space SQP methods.

Developer: Roscoe Bartlett

- Aristos: Optimization of large-scale design spaces
  - Invasive optimization approach
  - Based on full-space SQP methods
  - Efficiently manages inexactness in the inner linear solves

Developer: Denis Ridzal

Solver collaborations: Abstract interfaces and applications

Categories of Abstract Problems and Abstract Algorithms

Trilinos Packages

- Linear Problems: Given linear operator (matrix) $A \in \mathbf{R}^{n \times n}$
  - Linear equations: Solve $Ax = b$ for $x \in \mathbf{R}^n$ — **Belos**
  - Eigen problems: Solve $Av = \lambda v$ for (all) $v \in \mathbf{R}^n$ and $\lambda \in \mathbf{R}$ — **Anasazi**
- Nonlinear Problems: Given nonlinear operator $c(x, u) \in \mathbf{R}^{n+m} \rightarrow \mathbf{R}^n$
  - Nonlinear equations: Solve $c(x) = 0$ for $x \in \mathbf{R}^n$ — **NOX**
  - Stability analysis: For $c(x, u) = 0$ find space $u \in \mathcal{U}$ such that $\frac{\partial c}{\partial x}$ is singular — **LOCA**
- Transient Nonlinear 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 $x(t) \in \mathbf{R}^n, t \in [0, T]$ — **Rythmos**
- Optimization Problems:
  - Unconstrained: Find $u \in \mathbf{R}^n$ that minimizes $f(u)$ — **MOOCHO**
  - Constrained: Find $y \in \mathbf{R}^m$ and $u \in \mathbf{R}^n$ that: minimizes $f(y, u)$ such that $c(y, u) = 0$ — **Aristos**

Sandia National Laboratories

Abstract Numerical Algorithms

An **abstract numerical algorithm** (ANA) is a numerical algorithm that can be expressed solely in terms of vectors, vector spaces, and linear operators

Example Linear ANA (LANA) : Linear Conjugate Gradients

Given:

$A \in \mathcal{X} \to \mathcal{X}$: s.p.d. linear operator

$b \in \mathcal{X}$: right hand side vector

Find vector $x \in \mathcal{X}$ that solves $Ax = b$

- ANAs can be very mathematically sophisticated!
- ANAs can be extremely reusable!

Linear Conjugate Gradient Algorithm

Compute $r^{(0)} = b - Ax^{(0)}$ for the initial guess $x^{(0)}$.

for $i = 1, 2, \ldots$

$\rho_{i-1} = \left\langle r^{(i-1)}, r^{(i-1)} \right\rangle$

$\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$ $(\beta_0 = 0)$

$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)}$ $(p^{(1)} = r^{(1)})$

$q^{(i)} = Ap^{(i)}$

$\gamma_i = \left\langle p^{(i)}, q^{(i)} \right\rangle$

$\alpha_i = \rho_{i-1} / \gamma_i$

$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$

$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$

check convergence; continue if necessary

end

Types of operations

- linear operator applications
- vector-vector operations
- scalar operations
- scalar product <x,y> defined by vector space

Types of objects

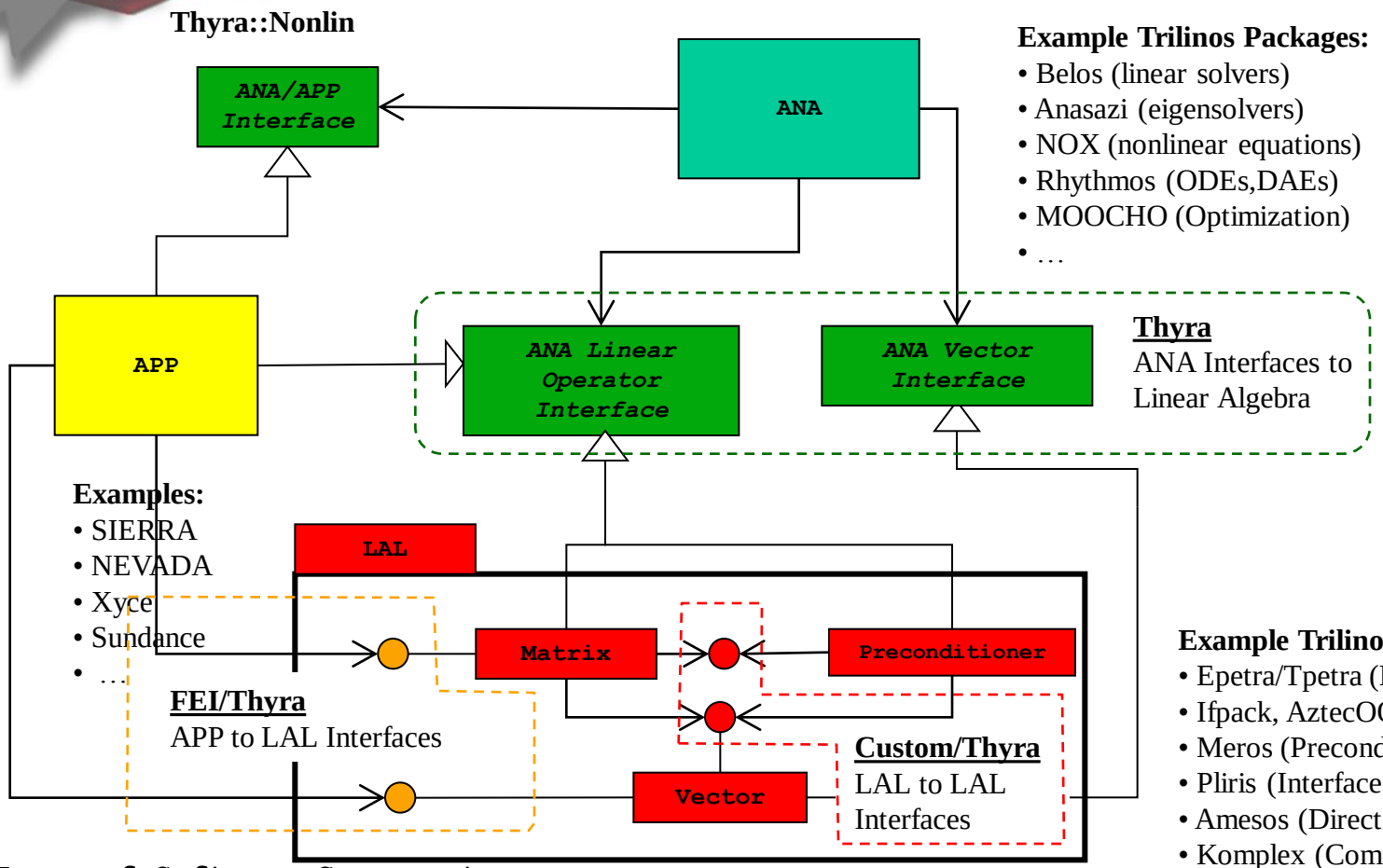
Linear Operators
- A

Vectors
- r, x, p, q

Scalars
- ρ, β, γ, α

Vector spaces?
- $\mathcal{X}$

Solver Software Components and Interfaces



Types of Software Components

1) ANA : Abstract Numerical Algorithm (e.g. linear solvers, eigensolvers, nonlinear solvers, stability analysis, uncertainty quantification, transient solvers, optimization etc.)

2) LAL : Linear Algebra Library (e.g. vectors, sparse matrices, sparse factorizations, preconditioners)

3) APP : Application (the model: physics, discretization method etc.)

Sandia National Laboratories

Introducing Stratimikos

- Greek **στρατηγική** (strategy) + **γραμμικός** (linear)
- Defines class Thyra::DefaultLinearSolverBuilder
- Uniform interface to many different:
  - Linear Solvers: Amesos, AztecOO, Belos, …
  - Preconditioners: Ifpack, ML, …
- Reads in options through a parameter list
  - Can change solver and its options at run time
  - Can read options from XML file
- Accepts any linear system objects that provide
  - Epetra_Operator / Epetra_RowMatrix view of the matrix
  - SPMD vector views for the right-hand side and initial guess vectors
    - e.g., Epetra_MultiVector

Stratimikos Parameter List and Sublists

```
<ParameterList name="Stratimikos">
  <Parameter name="Linear Solver Type" type="string" value="AztecOO"/>
  <Parameter name="Preconditioner Type" type="string" value="Ifpack"/>
  <ParameterList name="Linear Solver Types">
    <ParameterList name="Amesos">
      <Parameter name="Solver Type" type="string" value="Klu"/>
      <ParameterList name="Amesos Settings">
        <Parameter name="MatrixProperty" type="string" value="general"/>
        ...
        <ParameterList name="Mumps"> ... </ParameterList>
        <ParameterList name="Superludist"> ... </ParameterList>
      </ParameterList>
    </ParameterList>
    <ParameterList name="AztecOO">
      <ParameterList name="Forward Solve">
        <Parameter name="Max Iterations" type="int" value="400"/>
        <Parameter name="Tolerance" type="double" value="1e-06"/>
        <ParameterList name="AztecOO Settings">
          <Parameter name="Aztec Solver" type="string" value="GMRES"/>
          ...
        </ParameterList>
      </ParameterList>
      ...
    </ParameterList>
    <ParameterList name="Belos"> ... </ParameterList>
  </ParameterList>
<ParameterList name="Preconditioner Types">
    <ParameterList name="Ifpack">
      <Parameter name="Prec Type" type="string" value="ILU"/>
      <Parameter name="Overlap" type="int" value="0"/>
      <ParameterList name="Ifpack Settings">
        <Parameter name="fact: level-of-fill" type="int" value="0"/>
        ...
      </ParameterList>
    </ParameterList>
    <ParameterList name="ML"> ... </ParameterList>
  </ParameterList>
</ParameterList>
```

Top level parameters

Linear Solvers

Preconditioners

Sublists passed on to package code!

Every parameter and sublist is handled by Thyra code and is fully validated!

Sandia National Laboratories

Getting started: “How do I…?”

“How do I…?”

- Build my application with Trilinos?
- Learn about common Trilinos programming idioms?
- Download / find an installation of Trilinos?
- Find documentation and help?

Building your app with Trilinos

If you are using Makefiles:

- Makefile.export system

If you are using CMake:

- CMake FIND_PACKAGE

Using CMake to build with Trilinos

- CMake: Cross-platform build system
  - Similar function as the GNU Autotools
- Trilinos uses CMake to build
- You don't have to use CMake to build with Trilinos
- But if you do:
  - FIND_PACKAGE(Trilinos …)
  - Example CMake script in hands-on demo
- I find this much easier than hand-writing Makefiles

Export Makefile System

Once Trilinos is built, how do you link against the application?

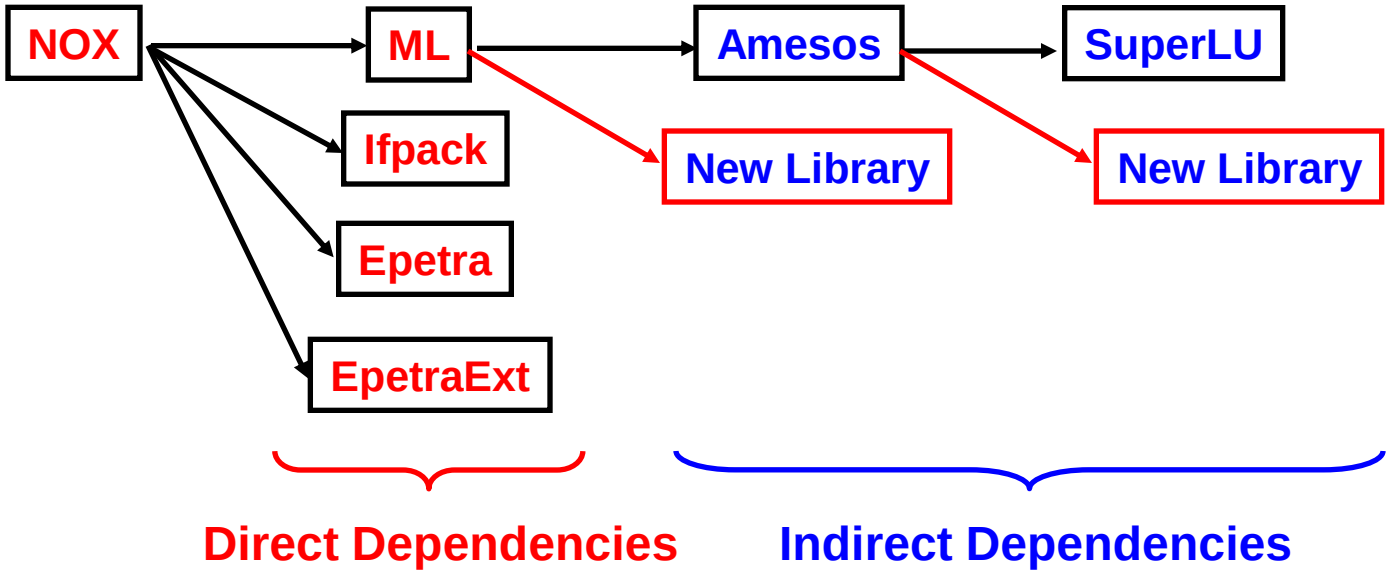
There are a number of issues:

- **Library link order:**
  - **-lnoxepetra -lnox –lepetra –lteuchos –lblas –llapack**
- **Consistent compilers:**
  - **g++, mpiCC, icc…**
- **Consistent build options and package defines:**
  - **g++ -g –O3 –D HAVE_MPI –D _STL_CHECKED**

Answer: Export Makefile system

Why Export Makefiles are Important

- Trilinos has LOTS of packages
- As package dependencies (especially optional ones) are introduced, more maintenance is required by the top-level packages:



NOX either must:

- Account for the new libraries in it's configure script (unscalable)
- Depend on direct dependent packages to supply them through export Makefiles

Export Makefiles in Action

```
#
# A Makefile that your application can use if you want to build with Epetra.
#
# (Excerpt from $(TRILINOS_INSTALL_DIR)/include/Makefile.client.Epetra.)
#

# Include the Trilinos export Makefile from package=Epetra.
include $(TRILINOS_INSTALL_DIR)/include/Makefile.export.Epetra

# Add the Trilinos installation directory to the library and header search paths.
LIB_PATH = $(TRILINOS_INSTALL_DIR)/lib
INCLUDE_PATH = $(TRILINOS_INSTALL_DIR)/include $(CLIENT_EXTRA_INCLUDES)

# Set the C++ compiler and flags to those specified in the export Makefile.
# This ensures your application is built with the same compiler and flags
# with which Trilinos was built.
CXX = $(EPETRA_CXX_COMPILER)
CXXFLAGS = $(EPETRA_CXX_FLAGS)

# Add the Trilinos libraries, search path, and rpath to the
# linker command line arguments
LIBS = $(CLIENT_EXTRA_LIBS) $(SHARED_LIB_RPATH_COMMAND) \
  $(EPETRA_LIBRARIES) \
  $(EPETRA_TPL_LIBRARIES) \
  $(EPETRA_EXTRA_LD_FLAGS)

#
# Rules for building executables and objects.
#
%.exe : %.o $(EXTRA_OBJS)
        $(CXX) -o $@ $(LDFLAGS) $(CXXFLAGS) $< $(EXTRA_OBJS) -L$(LIB_PATH) $(LIBS)

%.o : %.cpp
        $(CXX) -c -o $@ $(CXXFLAGS) -I$(INCLUDE_PATH) $(EPETRA_TPL_INCLUDES) $<
```

Software interface idioms

Idioms: Common “look and feel”

- Lower-level programming idioms
  - Provided by the Teuchos utilities package
  - Hierarchical “input deck” (ParameterList)
  - Memory management classes (RCP, ArrayRCP)
    - Safety: Manage data ownership and sharing
    - Performance: Avoid copies, control memory placement
  - Performance counters (e.g., TimeMonitor)
- Higher-level algorithmic idioms
  - Petra distributed object model
    - Provided by Epetra and Tpetra
    - Common “language” shared by many packages

ParameterList: Trilinos' "input deck"

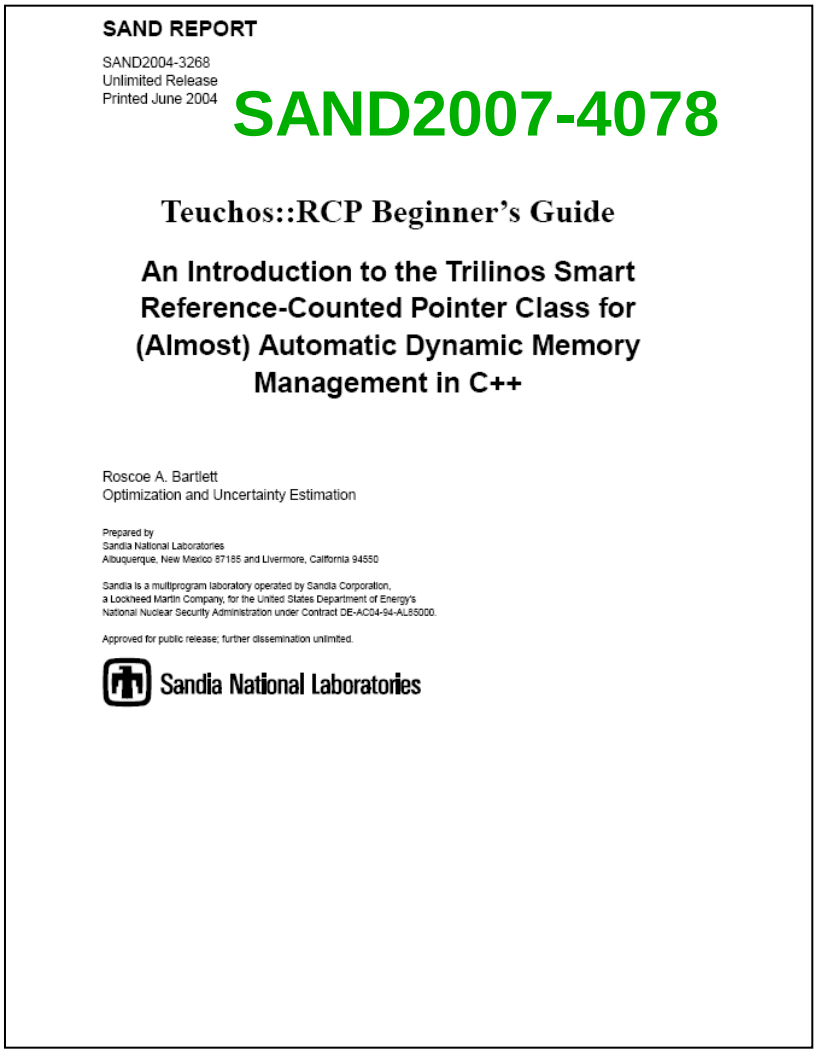
- Simple key/value pair database, but nest-able
  - Naturally hierarchical, just like numerical algorithms or software
  - Communication protocol between application layers
- Reproducible runs: save to XML, restore configuration
- Can express constraints and dependencies
- Optional GUI (Optika): lets novice users run your app

```
Teuchos::ParameterList p;
p.set("Solver", "GMRES");
p.set("Tolerance", 1.0e-4);
p.set("Max Iterations", 100);

Teuchos::ParameterList& lsParams = p.sublist("Solver Options");
lsParams.set("Fill Factor", 1);

double tol = p.get<double>("Tolerance");
int fill = p.sublist("Solver Options").get<int>("Fill Factor");
```

Memory management classes

- Scientific computation: Lots of data, big objects
  - Avoid copying and share data whenever possible
  - Who “owns” (deallocates) the data?
- Manual memory management (void*) not an option
  - Results in buggy and / or conservative code
- Reference-counted pointers (RCPs) and arrays
  - You don’t have to deallocate memory explicitly
  - Objects deallocated when nothing points to them anymore
- Important for multicore CPU and hybrid CPU/GPU!
  - Custom (de)allocators for GPU device memory
  - Avoid unnecessary data movement, preserve locality
    - CPU – GPU data transfers are expensive
    - Important for multicore CPU too (e.g., NUMA)

Teuchos::RCP Technical Report



SAND REPORT

SAND2004-3268
Unlimited Release
Printed June 2004

SAND2007-4078

Teuchos::RCP Beginner's Guide

An Introduction to the Trilinos Smart Reference-Counted Pointer Class for (Almost) Automatic Dynamic Memory Management in C++

Roscoe A. Bartlett
Optimization and Uncertainty Estimation

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Sandia National Laboratories

http://trilinos.sandia.gov/documentation.html

Trilinos/doc/RCPbeginnersGuide

Sandia National Laboratories

“But I don’t want RCPs!”

- They do add some keystrokes:
  - `RCP<Matrix>` vs. `Matrix*`
  - `ArrayRCP<double>` vs. `double[]`
- BUT: Run-time cost is none or very little
  - We have automated performance tests
- Debug build → useful error checking
  - More than Boost’s / C++0x’s shared_ptr
- Not every Trilinos package exposes them
  - Some packages hide them behind handles or typedefs
  - Python “skin” hides them; Python is garbage-collected
- RCPs part of interface between packages
  - Trilinos like LEGO™ blocks
  - Packages don’t have to worry about memory management
    - Easier for them to share objects in interesting ways

TimeMonitor

- Timers that keep track of:
  - Runtime
  - Number of calls
- Time object associates a string name to the timer:

```
RCP<Time> stuffTimer =
  TimeMonitor::getNewCounter ("Do Stuff");
```

- TimeMonitor guard controls timer in scope-safe way

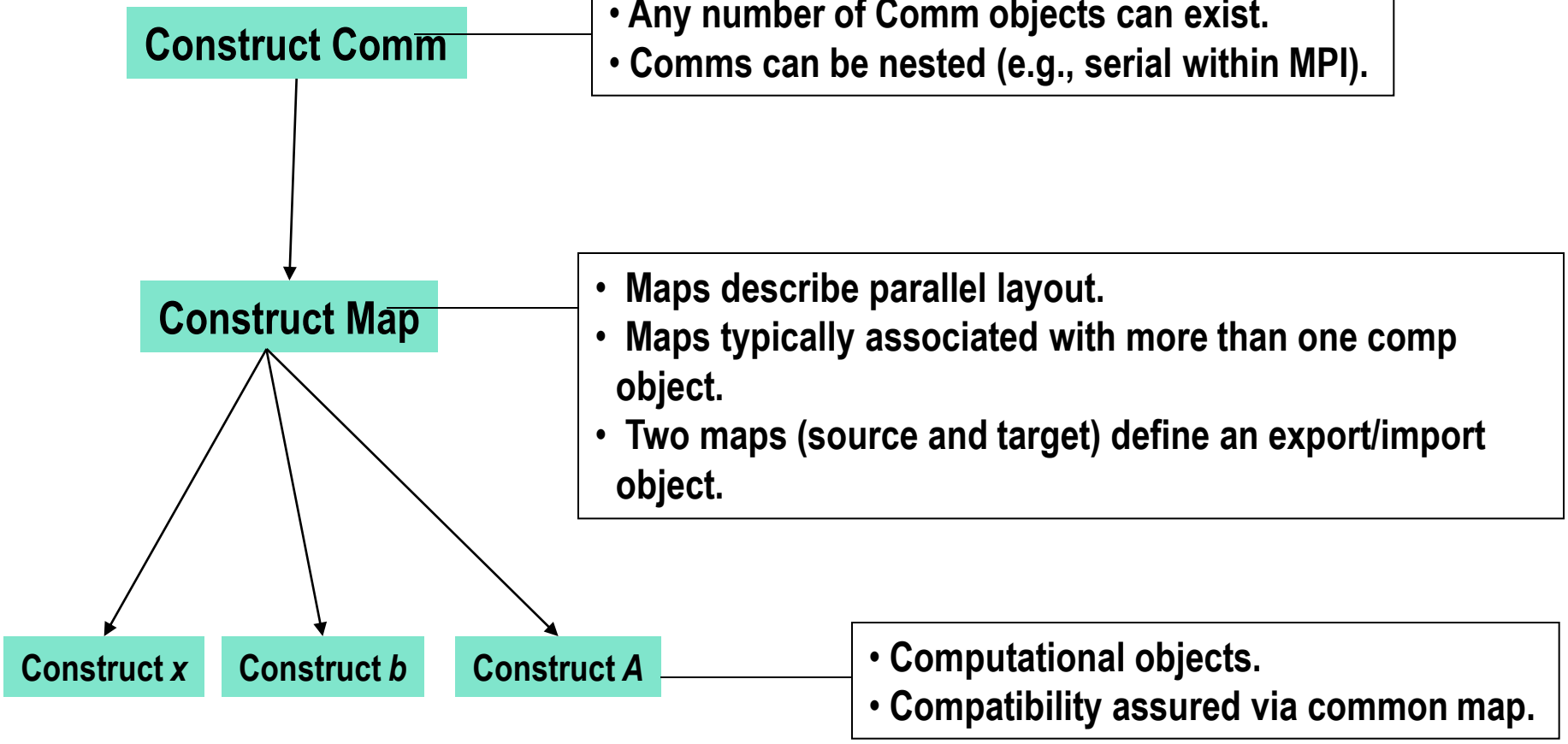
```
{
  TimeMonitor tm (*stuffTimer);
  doStuff ();
}
```

- Automatically takes care of recursive / nested calls
- Scalable, safe parallel timer statistics summary
  - `TimeMonitor::summarize ();`

Petra Distributed Object Model

Sandia National Laboratories

Typical Petra Object Construction Sequence



Sandia National Laboratories

Petra Implementations

- Epetra (Essential Petra):
  - Current production version
  - Uses stable core subset of C++ (circa 2000)
  - Restricted to real, double precision arithmetic
  - Interfaces accessible to C and Fortran users

- Tpetra (Templated Petra):
  - Next-generation version
  - Needs a modern C++ compiler (but not C++0x)
  - Supports arbitrary scalar and index types via templates
    - Arbitrary- and mixed-precision arithmetic
    - 64-bit indices for solving problems with >2 billion unknowns
  - Hybrid MPI / shared-memory parallel
    - Supports multicore CPU and hybrid CPU/GPU
    - Built on Kokkos manycore node library

Developers: Chris Baker, Mike Heroux, Rob Hoekstra, Alan Williams

A Simple Epetra/AztecOO Program

```
// Header files omitted…
int main(int argc, char *argv[]) {
Epetra_SerialComm Comm();
```

```
// ***** Map puts same number of equations on each pe *****

  int NumMyElements = 1000 ;
  Epetra_Map Map(-1, NumMyElements, 0, Comm);
  int NumGlobalElements = Map.NumGlobalElements();
```

```
// ***** Create an Epetra_Matrix  tridiag(-1,2,-1) *****

  Epetra_CrsMatrix A(Copy, Map, 3);
  double negOne = -1.0; double posTwo = 2.0;

  for (int i=0; i<NumMyElements; i++) {
    int GlobalRow = A.GRID(i);
    int RowLess1 = GlobalRow - 1;
    int RowPlus1 = GlobalRow + 1;
    if (RowLess1!=-1)
      A.InsertGlobalValues(GlobalRow, 1, &negOne, &RowLess1);
    if (RowPlus1!=NumGlobalElements)
      A.InsertGlobalValues(GlobalRow, 1, &negOne, &RowPlus1);
    A.InsertGlobalValues(GlobalRow, 1, &posTwo, &GlobalRow);
  }
A.FillComplete(); // Transform from GIDs to LIDs
```

```
  // ***** Create x and b vectors *****
  Epetra_Vector x(Map);
  Epetra_Vector b(Map);
  b.Random(); // Fill RHS with random #s
```

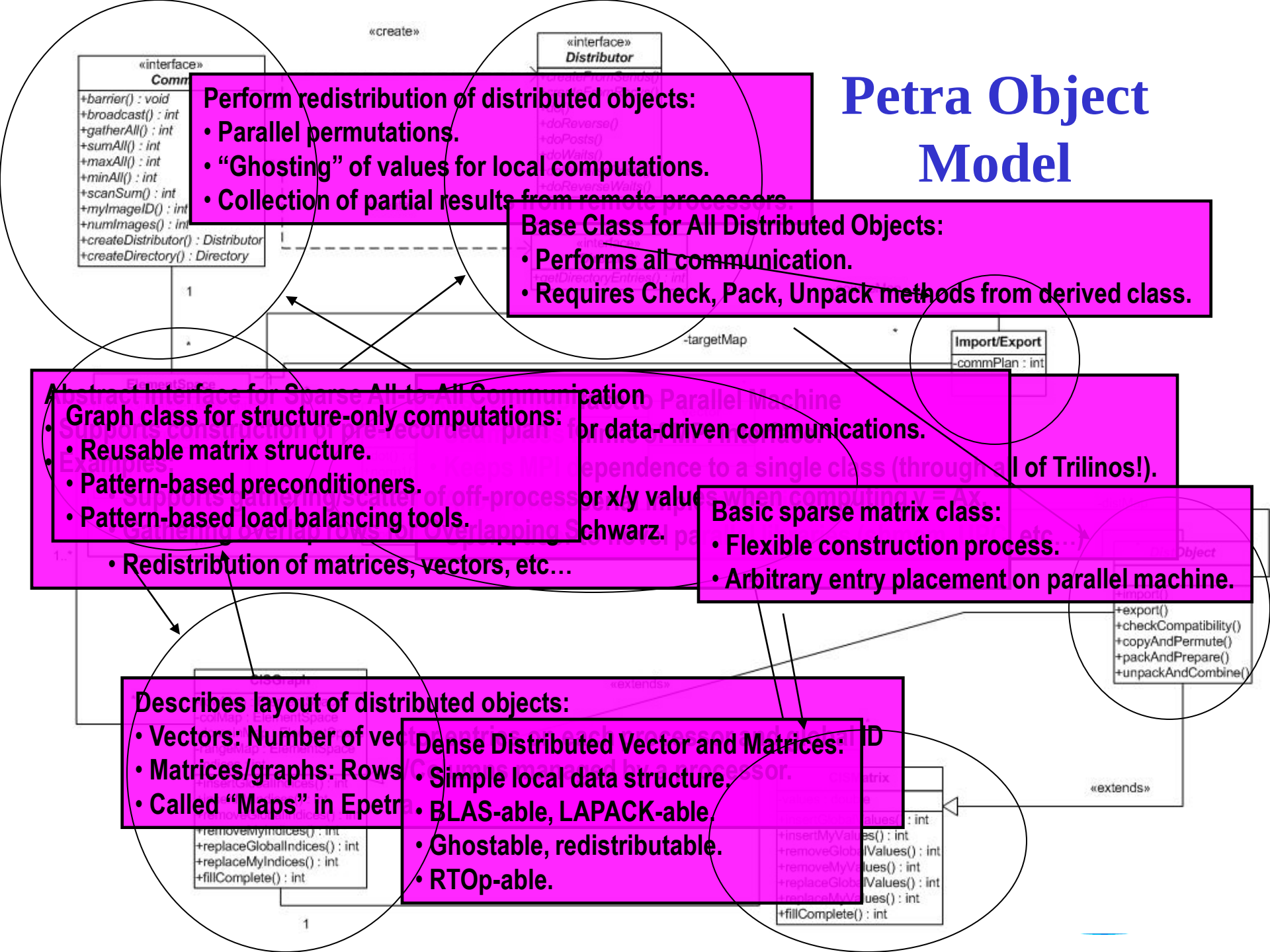
```
// ***** Create Linear Problem *****
  Epetra_LinearProblem problem(&A, &x, &b);
```

```
  // ***** Create/define AztecOO instance, solve *****
  AztecOO solver(problem);
  solver.SetAztecOption(AZ_precond, AZ_Jacobi);
  solver.Iterate(1000, 1.0E-8);
```

```
// ***** Report results, finish **********************
  cout << "Solver performed " << solver.NumIters()
       << " iterations." << endl
       << "Norm of true residual = "
       << solver.TrueResidual()
       << endl;


  return 0;
}
```

Sandia National Laboratories

Petra Object Model



Perform redistribution of distributed objects:
- Parallel permutations.
- "Ghosting" of values for local computations.
- Collection of partial results from remote processors.

Base Class for All Distributed Objects:
- Performs all communication.
- Requires Check, Pack, Unpack methods from derived class.

Graph class for structure-only computations:
- Reusable matrix structure.
- Pattern-based preconditioners.
- Pattern-based load balancing tools.

- Redistribution of matrices, vectors, etc…

Basic sparse matrix class:
- Flexible construction process.
- Arbitrary entry placement on parallel machine.

Describes layout of distributed objects:
- Vectors: Number of ve
- Matrices/graphs: Rows
- Called "Maps" in Epetr

Dense Distributed Vector and Matrices:
- Simple local data structure.
- BLAS-able, LAPACK-able.
- Ghostable, redistributable.
- RTOp-able.

Details about Epetra Maps

- Note: Focus on Maps (not BlockMaps).
- Getting beyond standard use case…

- **Note: All of the concepts presented here for Epetra carry over to Tpetra!**

1-to-1 Maps

- A map is 1-to-1 if…
  - Each global ID appears only once in the map
  - (and is thus associated with only a single processor)
- Certain operations in parallel data repartitioning require 1-to-1 maps:
  - Source map of an import must be 1-to-1.
  - Target map of an export must be 1-to-1.
  - Domain map of a 2D object must be 1-to-1.
  - Range map of a 2D object must be 1-to-1.

2D Objects: Four Maps

- Epetra 2D objects:
  - CrsMatrix, FECrsMatrix
  - CrsGraph
  - VbrMatrix, FEVbrMatrix
- Have four maps:
  - **RowMap**: On each processor, the global IDs of the **rows** that processor will "manage."
  - **ColMap**: On each processor, the global IDs of the **columns** that processor will "manage."
  - **DomainMap**: The layout of domain objects (the *x* (multi)vector in $y = Ax$).
  - **RangeMap**: The layout of range objects (the *y* (multi)vector in $y = Ax$).

Typically a 1-to-1 map

Typically NOT a 1-to-1 map

Must be 1-to-1 maps!!!

Sandia National Laboratories

Sample Problem

$$\underset{\boldsymbol{y}}{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}} = \underset{\boldsymbol{A}}{\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}} \underset{\boldsymbol{x}}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}$$

Sandia National Laboratories

Case 1: Standard Approach

- First 2 rows of *A*, elements of *y* and elements of *x*, kept on PE 0.
- Last row of *A*, element of *y* and element of *x*, kept on PE 1.

PE 0 Contents

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \ldots A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \end{bmatrix}, \ldots x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- RowMap = {0, 1}
- ColMap = {0, 1, 2}
- DomainMap = {0, 1}
- RangeMap = {0, 1}

PE 1 Contents

$$y = \begin{bmatrix} y_3 \end{bmatrix}, \ldots A = \begin{bmatrix} 0 & -1 & 2 \end{bmatrix}, \ldots x = \begin{bmatrix} x_3 \end{bmatrix}$$

- RowMap = {2}
- ColMap = {1, 2}
- DomainMap = {2}
- RangeMap = {2}

Original Problem

$$\begin{matrix} y \\ \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \end{matrix} = \begin{matrix} A \\ \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \end{matrix} \begin{matrix} x \\ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \end{matrix}$$

Notes:

- Rows are wholly owned.
- RowMap=DomainMap=RangeMap (all 1-to-1).
- ColMap is NOT 1-to-1.
- Call to FillComplete: A.FillComplete(); // Assumes

Sandia National Laboratories

Case 2: Twist 1

- First 2 rows of *A*, first element of *y* and last 2 elements of *x*, kept on PE 0.
- Last row of *A*, last 2 element of *y* and first element of *x*, kept on PE 1.

PE 0 Contents

$$y = [y_1], \ldots A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \end{bmatrix}, \ldots x = \begin{bmatrix} x_2 \\ x_3 \end{bmatrix}$$

- RowMap = {0, 1}
- ColMap = {0, 1, 2}
- DomainMap = {1, 2}
- RangeMap = {0}

PE 1 Contents

$$y = \begin{bmatrix} y_2 \\ y_3 \end{bmatrix}, \ldots A = [0 \quad -1 \quad 2], \ldots x = [x_1]$$

- RowMap = {2}
- ColMap = {1, 2}
- DomainMap = {0}
- RangeMap = {1, 2}

Original Problem

$$\begin{matrix} y \\ \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \end{matrix} = \begin{matrix} A \\ \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \end{matrix} \begin{matrix} x \\ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \end{matrix}$$

Notes:

- Rows are wholly owned.
- RowMap is NOT = DomainMap
  is NOT = RangeMap (all 1-to-1).
- ColMap is NOT 1-to-1.
- Call to FillComplete:
  A.FillComplete(DomainMap, RangeMap);

Sandia National Laboratories

Case 2: Twist 2

- First row of *A*, part of second row of *A*, first element of *y* and last 2 elements of *x*, kept on PE 0.
- Last row, part of second row of *A*, last 2 element of *y* and first element of *x*, kept on PE 1.

PE 0 Contents

$$y = [y_1], \ldots A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 1 & 0 \end{bmatrix}, \ldots x = \begin{bmatrix} x_2 \\ x_3 \end{bmatrix}$$

- RowMap = {0, 1}
- ColMap = {0, 1}
- DomainMap = {1, 2}
- RangeMap = {0}

PE 1 Contents

$$y = \begin{bmatrix} y_2 \\ y_3 \end{bmatrix}, \ldots A = \begin{bmatrix} 0 & 1 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \ldots x = [x_1]$$

- RowMap = {1, 2}
- ColMap = {1, 2}
- DomainMap = {0}
- RangeMap = {1, 2}

Original Problem

$$\begin{matrix} y & & A & & x \end{matrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Notes:

- Rows are NOT wholly owned.
- RowMap is NOT = DomainMap
  is NOT = RangeMap (all 1-to-1).
- RowMap and ColMap are NOT 1-to-1.
- Call to FillComplete:
  A.FillComplete(DomainMap, RangeMap);

Sandia National Laboratories

What does FillComplete do?

- Signals you're done defining matrix structure
- Does a bunch of stuff
- e.g., create import/export objects (if needed)for distributed sparse matrix-vector multiply:
  - If ColMap ≠ DomainMap, create Import object
  - If RowMap ≠ RangeMap, create Export object
- A few rules:
  - Non-square matrices will *always* require:

    ```
    A.FillComplete(DomainMap,RangeMap);
    ```

  - DomainMap and RangeMap *must be 1-to-1*

How do I learn more?

Sandia National Laboratories

How do I learn more?

- Documentation:
  - Trilinos tutorial: http://trilinos.sandia.gov/Trilinos10.6Tutorial.pdf
  - Per-package documentation: http://trilinos.sandia.gov/packages/
  - Trilinos Wiki with more examples: https://code.google.com/p/trilinos/wiki/
- E-mail lists:
  - http://trilinos.sandia.gov/mail_lists.html
- Annual user meetings and tutorials:
  - DOE ACTS Tutorial (here we are!)
  - Trilinos User Group (TUG) meeting and tutorial
    - First week of November at SNL / NM
    - Talks available for download (slides and video):
      - http://trilinos.sandia.gov/events/trilinos_user_group_2010
      - http://trilinos.sandia.gov/events/trilinos_user_group_2009
      - http://trilinos.sandia.gov/events/trilinos_user_group_2008
  - NEW! “EuroTUG” (in Europe)
    - Planned for the first week of June 2012

How do I get Trilinos?

- Current release (10.6) available for download:
  - http://trilinos.sandia.gov/download/trilinos-10.6.html
  - Source tarball with sample build scripts
- Cray packages recent releases of Trilinos
  - http://www.nersc.gov/users/software/programming-libraries/math-libraries/trilinos/
  - `$ module load trilinos`
- LGPL or BSD license (depending on the package)

How do I build Trilinos?

- Need C and C++ compiler and the following tools:
  - CMake (version >= 2.8)
  - (Optimized) LAPACK and BLAS
- Optional software:
  - MPI library (for distributed-memory computation)
  - Many other third-party libraries
- You may need to write a short configure script
  - Sample configure scripts in sampleScripts/
  - Find one closest to your software setup, & tweak it
- Build sequence looks like GNU Autotools
  1. Invoke your configure script, that invokes CMake
  2. `make`
  3. `make install`
- Documentation:
  - http://trilinos.sandia.gov/Trilinos10CMakeQuickstart.txt
  - Ask me at the hands-on if interested

Hands-on tutorial

- Trilinos Wiki
  - https://code.google.com/p/trilinos/wiki/TrilinosHandsOnTutorial
  - Example codes: https://code.google.com/p/trilinos/w/list
  - All examples are working codes
    - Tested with Trilinos 10.4 and 10.7 (development branch)
- Web interface to Trilinos
  - https://www.users.csbsju.edu/trilinos/WebTrilinosMPI/c++/index.html
  - Development branch of Trilinos (10.7)
  - Need username and password (will give these out later)
  - All you need is a web browser!
    - Copy, paste, and edit code examples in box
- Trilinos is also installed on NERSC machines
- If there is interest, I'll help install it on your machine too

Any questions?

Extra Slides

External Visibility

- Awards: R&D 100, HPC SW Challenge (04).
- www.cfd-online.com:

Trilinos 

A project led by Sandia to develop an object-oriented software framework for scientific computations. This is an active project which includes several state-of-the-art solvers and lots of other nice things a software engineer writing CFD codes would find useful. Everything is freely available for download once you have registered. Very good!

- Industry Collaborations: Various.
- Linux distros: Debian, Mandriva, Ubuntu, Fedora.
- SciDAC TOPS-2 partner, EASI (with ORNL, UT-Knoxville, UIUC, UC-Berkeley).
- Over 10,000 downloads since March 2005.
- Occasional unsolicited external endorsements such as the following two-person exchange on mathforum.org:

> The consensus seems to be that OO has little, if anything, to offer
> (except bloat) to numerical computing.
I would completely disagree. A good example of using OO in numerics is
Trilinos: http://software.sandia.gov/trilinos/

Trilinos / PETSc Interoperability

- Epetra_PETScAIJMatrix class
  - Derives from Epetra_RowMatrix
  - Wrapper for serial/parallel PETSc aij matrices
  - Utilizes callbacks for matrix-vector product, getrow
  - No deep copies
- Enables PETSc application to construct and call virtually any Trilinos preconditioner
- ML accepts fully constructed PETSc KSP solvers as smoothers
  - Fine grid only
  - Assumes fine grid matrix is really PETSc aij matrix
- Complements Epetra_PETScAIJMatrix class
  - For any smoother with getrow kernel, PETSc implementation should be *much* faster than Trilinos
  - For any smoother with matrix-vector product kernel, PETSc and Trilinos implementations should be comparable

Linear System Solves

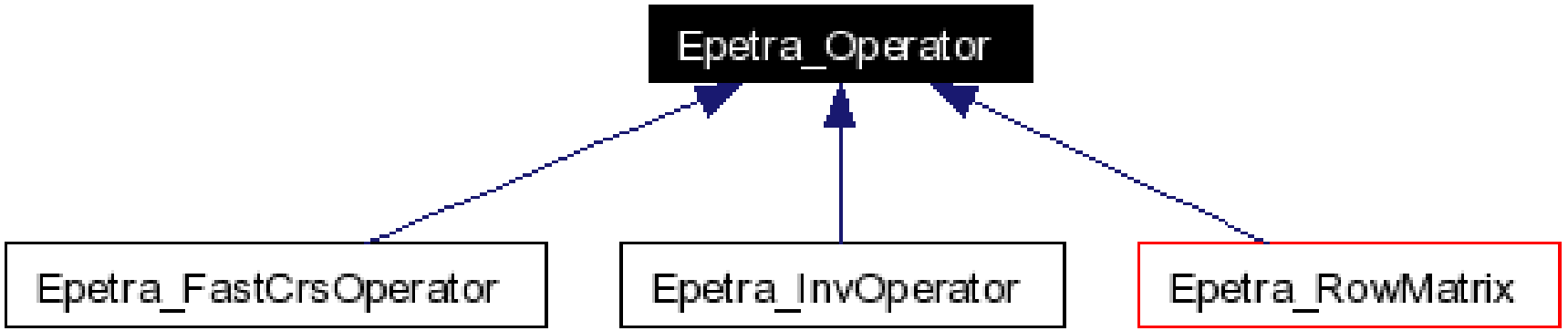
AztecOO

- Aztec is the previous workhorse solver at Sandia:
  - Extracted from the MPSalsa reacting flow code.
  - Installed in dozens of Sandia apps.
- AztecOO leverages the investment in Aztec:
  - Uses Aztec iterative methods and preconditioners.
- AztecOO improves on Aztec by:
  - Using Epetra objects for defining matrix and RHS.
  - Providing more preconditioners/scalings.
  - Using C++ class design to enable more sophisticated use.
- AztecOO interfaces allows:
  - Continued use of Aztec for functionality.
  - Introduction of new solver capabilities outside of Aztec.
- Belos is coming along as alternative.
  - AztecOO will not go away.
  - Will encourage new efforts and refactorings to use Belos.

AztecOO Extensibility

- AztecOO is designed to accept externally defined:
  - **Operators** (both *A* and *M*):
    - The linear operator *A* is accessed as an Epetra_Operator.
    - Users can register a preconstructed preconditioner as an Epetra_Operator.
  - **RowMatrix**:
    - If *A* is registered as a RowMatrix, Aztec's preconditioners are accessible.
    - Alternatively *M* can be registered separately as an Epetra_RowMatrix, and Aztec's preconditioners are accessible.
  - **StatusTests**:
    - Aztec's standard stopping criteria are accessible.
    - Can override these mechanisms by registering a StatusTest Object.

AztecOO understands Epetra_Operator



- AztecOO is designed to accept externally defined:
  - Operators (both *A* and *M*).
  - RowMatrix (Facilitates use of AztecOO preconditioners with external *A*).
  - StatusTests (externally-defined stopping criteria).

Belos and Anasazi

- Next generation linear solver / eigensolver library, written in templated C++.
- Provide a generic interface to a collection of algorithms for solving large-scale linear problems / eigenproblems.
- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - e.g.: MultiVecTraits, OperatorTraits
  - e.g.: SolverManager, Eigensolver / Iteration, Eigenproblem/ LinearProblem, StatusTest, OrthoManager, OutputManager
- Includes block linear solvers / eigensolvers:
  - Higher operator performance.
  - More reliable.
- Solves:
  - $AX = X\Lambda$ or $AX = BX\Lambda$ (Anasazi)
  - $AX = B$ (Belos)

Why are Block Solvers Useful?

- Block Solvers (in general):
  - Achieve better performance for operator-vector products.
- Block Eigensolvers (Op(A)X = LX):
  - Reliably determine multiple and/or clustered eigenvalues.
  - Example applications: Modal analysis, stability analysis, bifurcation analysis (LOCA)
- Block Linear Solvers (Op(A)X = B):
  - Useful for when multiple solutions are required for the same system of equations.
  - Example applications:
    - Perturbation analysis
    - Optimization problems
    - Single right-hand sides where A has a handful of small eigenvalues
    - Inner-iteration of block eigensolvers

Linear / Eigensolver Software Design

Belos and Anasazi are solver libraries that:

1. Provide an abstract interface to an operator-vector products, scaling, and preconditioning.
2. Allow the user to enlist any linear algebra package for the elementary vector space operations essential to the algorithm. (Epetra, PETSc, etc.)
3. Allow the user to define convergence of any algorithm (a.k.a. status testing).
4. Allow the user to determine the verbosity level, formatting, and processor for the output.
5. Allow these decisions to be made at runtime.
6. Allow for easier creation of new solvers through “managers” using “iterations” as the basic kernels.

Nonlinear System Solves

NOX/LOCA: Nonlinear Solver and Analysis Algorithms

NOX and LOCA are a combined package for solving and analyzing sets of nonlinear equations.

- NOX: Globalized Newton-based solvers.
- LOCA: Continuation, Stability, and Bifurcation Analysis.

We define the nonlinear problem:

$$\text{given } F : \mathbb{R}^n \rightarrow \mathbb{R}^n,$$

$$\text{find } x_* \in \mathbb{R}^n \text{ such that } F(x_*) = 0 \in \mathbb{R}^n$$

F is the residual or function evaluation

x is the solution vector

$J \in \mathbb{R}^{n \times n}$ is the Jacobian Matrix defined by: $J_{ij} = \dfrac{\partial F_i}{\partial x_j}$

Nonlinear Solver Algorithms

Tensor Method

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

Newton's Method

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

Broyden's Method

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

Globalizations

Line Search
Interval Halving
Quadratic
Cubic
More'-Thuente
Curvilinear (Tensor)

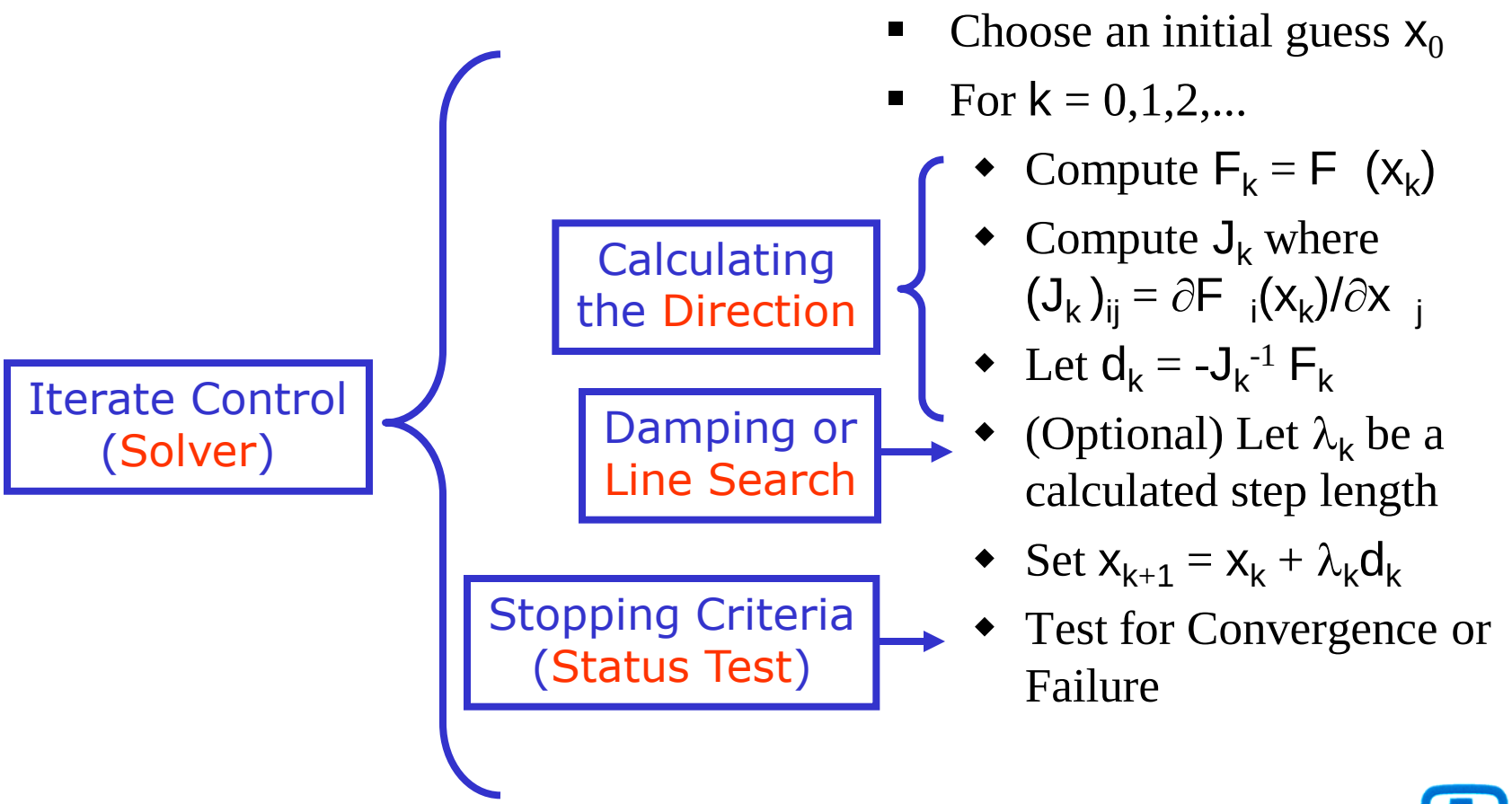
Trust Region
Dogleg
Inexact Dogleg

Homotopy
Artificial Parameter Continuation
Natural Parameter Continuation

Iterative Linear Solvers: Adaptive Forcing Terms
Jacobian-Free Newton-Krylov
Jacobian Estimation: Colored Finite Difference

Building Blocks of NOX

Example: Newton's Method for $F(x) = 0$



Sandia National Laboratories

Stopping Criteria
(StatusTests)

Highly Flexible Design: Users build a convergence test hierarchy and registers it with the solver (via solver constructor or reset method).

- Norm F: {Inf, One, Two} {absolute, relative} $\|F\| \leq \text{tol}$
- Norm Update ΔX: {Inf, One, Two} $\|x_k - x_{k-1}\| \leq \text{tol}$
- Norm Weighted Root Mean Square (WRMS):

$$C\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i^k - x_i^{k-1}}{\text{RTOL}|x_i^{k-1}| + \text{ATOL}_i}\right)^2} \leq \text{tol}$$

- Max Iterations: Failure test if solver reaches max # iters
- FiniteValue: Failure test that checks for NaN and Inf on $\|F\|$
- Stagnation: Failure test that triggers if the convergence rate fails a tolerance check for n consecutive iterations.

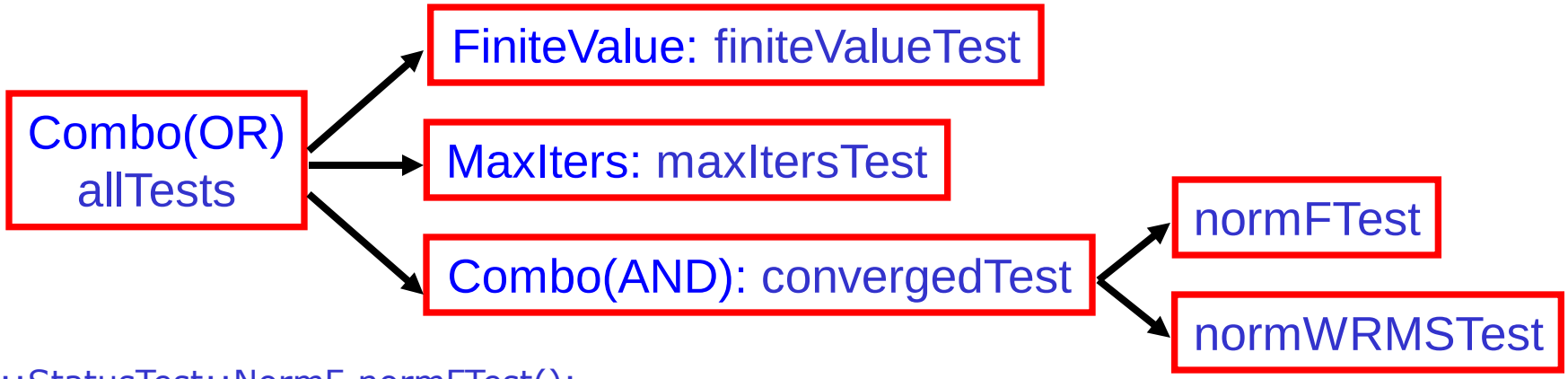
$$\frac{\|F_k\|}{\|F_{k-1}\|} \geq \text{tol}$$

- Combination: {AND, OR}
- Users Designed: Derive from NOX::StatusTest::Generic

Sandia National Laboratories

Building a Status Test

- Converge if both: $\|F\| \leq 1.0E-6$ $\|\delta x\|_{WRMS} \leq 1.0$
- Fail if value of $\|F\|$ becomes Nan or Inf
- Fail if we reach maximum iterations




```
NOX::StatusTest::NormF normFTest();
NOX::StatusTest::NormWRMS normWRMSTest();
NOX::StatusTest::Combo convergedTest(NOX::StatusTest::Combo::AND);
convergedTest.addStatusTest(normFTest);
convergedTest.addStatusTest(normWRMSTest);
NOX::StatusTest::FiniteValue finiteValueTest;
NOX::StatusTest::MaxIters maxItersTest(200);
NOX::StatusTest::Combo allTests(NOX::StatusTest::Combo::OR);
allTests.addStatusTest(finiteValueTest);
allTests.addStatusTest(maxItersTest);
allTests.addStatusTest(convergedTest);
```

Sandia National Laboratories

Status Tests Continued

```
-- Final Status Test Results --
Converged....OR Combination ->
  Converged....AND Combination ->
    Converged....F-Norm = 3.567e-13 < 1.000e-08
                  (Length-Scaled Two-Norm, Absolute Tolerance)
    Converged....WRMS-Norm = 1.724e-03 < 1
                  (Min Step Size:  1.000e+00 >= 1)
                  (Max Lin Solv Tol:  4.951e-14 < 0.5)
  ??..........Finite Number Check (Two-Norm F) = Unknown
  ??..........Number of Iterations = -1 < 200
```

User Defined are Derived from NOX::StatusTest::Generic

```
NOX::StatusTest::StatusType checkStatus(const NOX::Solver::Generic &problem)

NOX::StatusTest::StatusType
checkStatusEfficiently(const NOX::Solver::Generic &problem,
                       NOX::StatusTest::CheckType checkType)

NOX::StatusTest::StatusType getStatus() const

ostream& print(ostream &stream, int indent=0) const
```

NOX Interface

NOX solver methods are ANAs, and are implemented in terms of group/vector abstract interfaces:

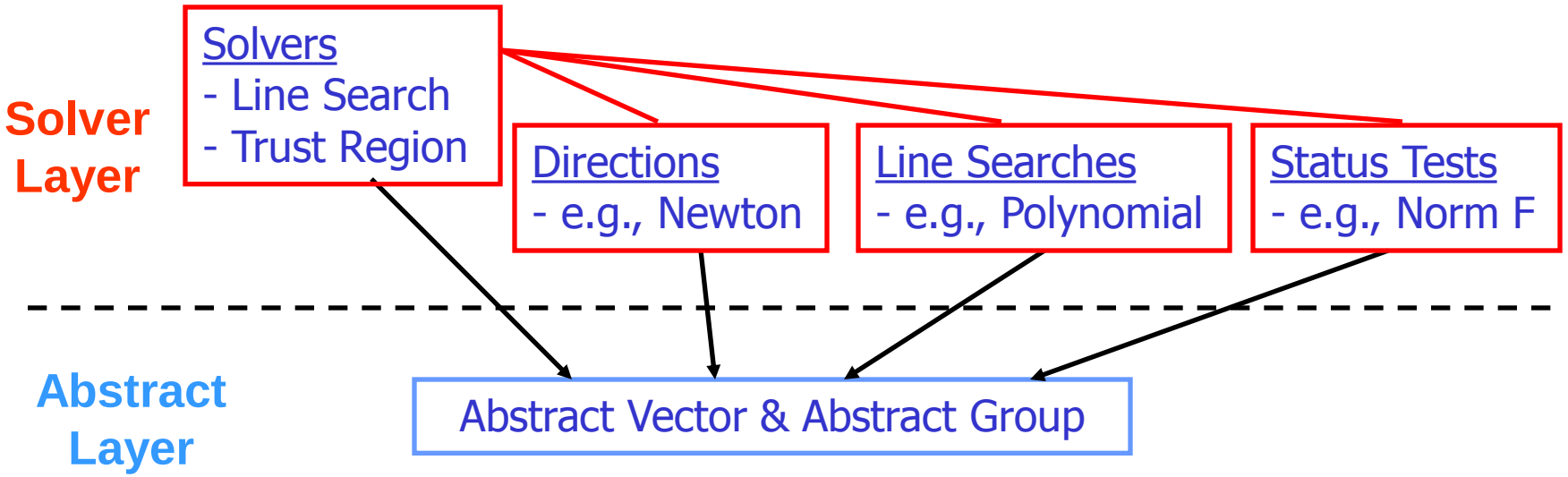
Group	Vector
`computeF()`	`innerProduct()`
`computeJacobian()`	`scale()`
`applyJacobianInverse()`	`norm()`
	`update()`

NOX solvers will work with any group/vector that implements these interfaces.

Four concrete implementations are supported:

1. LAPACK
2. EPETRA
3. PETSc
4. Thyra (Release 8.0)

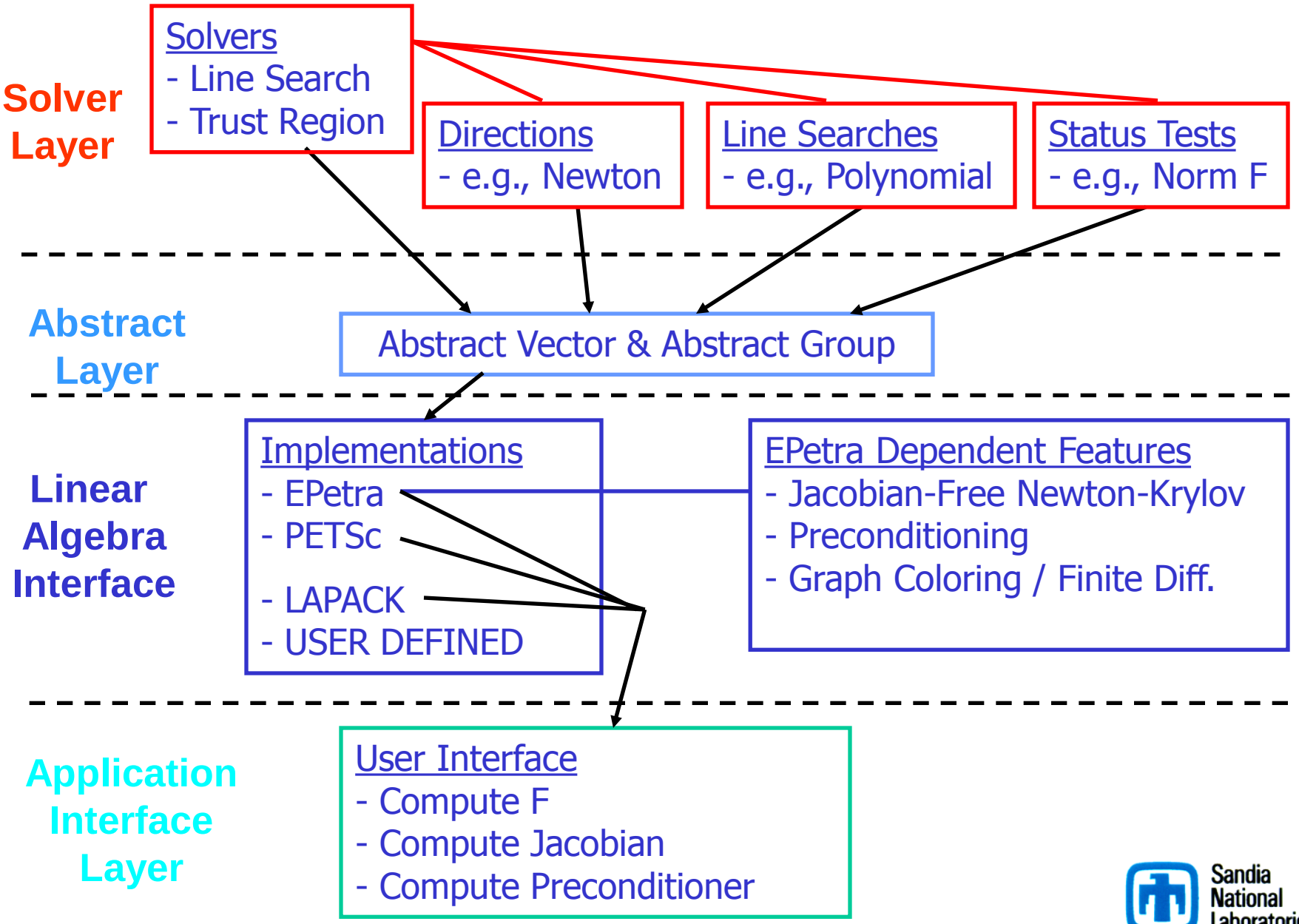
NOX Interface



- Don't need to directly access the vector or matrix entries, only manipulate the objects.
- NOX uses an abstract interface to manipulate linear algebra objects.
- Isolate the Solver layer from the linear algebra implementations used by the application.
- This approach means that NOX does NOT rely on any specific linear algebra format.
- Allows the apps to tailor the linear algebra to their own needs!
  - Serial or Parallel
  - Any Storage format: User Defined, LAPACK, PETSc, Epetra

Sandia National Laboratories

NOX Framework



Sandia National Laboratories

The Epetra "Goodies"

- Matrix-Free Newton-Krylov Operator
  - Derived from Epetra_Operator
  - Can be used to estimate Jacobian action on a vector
  - NOX::Epetra::MatrixFree

$$Jy = \frac{F(x + y\delta) - F(x)}{\delta}$$

- Finite Difference Jacobian
  - Derived from an Epetra_RowMatrix
  - Can be used as a preconditioner matrix
  - NOX::Epetra::FiniteDifference

$$J_j = \frac{F(x + \delta e_j) - F(x)}{\delta}$$

- Graph Colored Finite Difference Jacobian
  - Derived from NOX::Epetra::FiniteDifference
  - Fast Jacobian fills – need connectivity/coloring graph
  - (NOX::Epetra::FiniteDifferenceColoring)

- Full interface to AztecOO using NOX parameter list
- Preconditioners: internal AztecOO, Ifpack, User defined
- Scaling object

Trilinos Awards

- 2004 R&D 100 Award
- SC2004 HPC Software Challenge Award
- Sandia Team Employee Recognition Award
- Lockheed-Martin Nova Award Nominee

EpetraExt: Extensions to Epetra

- Library of useful classes not needed by everyone
- Most classes are types of “transforms”.
- Examples:
  - Graph/matrix view extraction.
  - Epetra/Zoltan interface.
  - Explicit sparse transpose.
  - Singleton removal filter, static condensation filter.
  - Overlapped graph constructor, graph colorings.
  - Permutations.
  - Sparse matrix-matrix multiply.
  - Matlab, MatrixMarket I/O functions.
- Most classes are small, useful, but non-trivial to write.

Developer: Robert Hoekstra, Alan Williams, Mike Heroux

Trilinos Strategic Goals

- **Scalable Computations:** As problem size and processor counts increase, the cost of the computation will remain nearly fixed.
- **Hardened Computations:** Never fail unless problem essentially intractable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.
- **Full Vertical Coverage:** Provide leading edge enabling technologies through the entire technical application software stack: from problem construction, solution, analysis and optimization.

Algorithmic Goals

- ***Grand* Universal Interoperability**: All Trilinos packages, and important external packages, will be interoperable, so that any combination of packages and external software (e.g., PETSc, Hypre) that makes sense algorithmically will be possible within Trilinos.
- **Universal Accessibility:** All Trilinos capabilities will be available to users of major computing environments: C++, Fortran, Python and the Web, and from the desktop to the latest scalable systems.
- **Universal Solver RAS**: Trilinos will be:
  - Reliable: Leading edge hardened, scalable solutions for each of these applications
  - Available: Integrated into every major application at Sandia
  - Serviceable: Easy to maintain and upgrade within the application environment.

Software Goals

Highlights from some Trilinos packages

- Amesos2 (sparse direct solvers interface)
- Belos (iterative linear solvers)
- Ifpack2 (incomplete factorizations)
- Intrepid (PDE discretizations)
- Kokkos (manycore API and kernels)
- MueLu (manycore-friendly multigrid)
- ShyLU (new direct / iterative hybrid solver)