An Overview of Trilinos

Michael A. Heroux
Sandia National Laboratories

SAND 2009-3732P

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

Outline of Talk

- Background / Motivation / Evolution.
- Trilinos Package Concepts.
- Whirlwind Tour of Trilinos Packages.
- Getting Started.
- Solver Collaborations: ANAs, LALs and APPs.
- Concluding remarks.

- Hands On Tutorial

Trilinos Contributors

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

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

Past Contributors

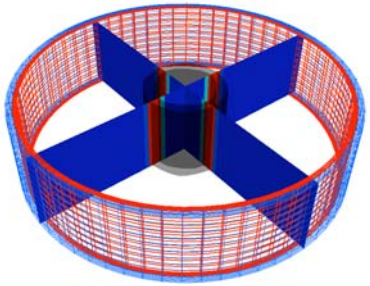
Jason Cross
Michael Gee
Esteban Guillen
Bob Heaphy
Kris Kampshoff
Ian Karlin
Sarah Knepper
Tammy Kolda
Joe Outzen
Mike Phenow
Paul Sexton
Bob Shuttleworth
Ken Stanley

Sandia National Laboratories

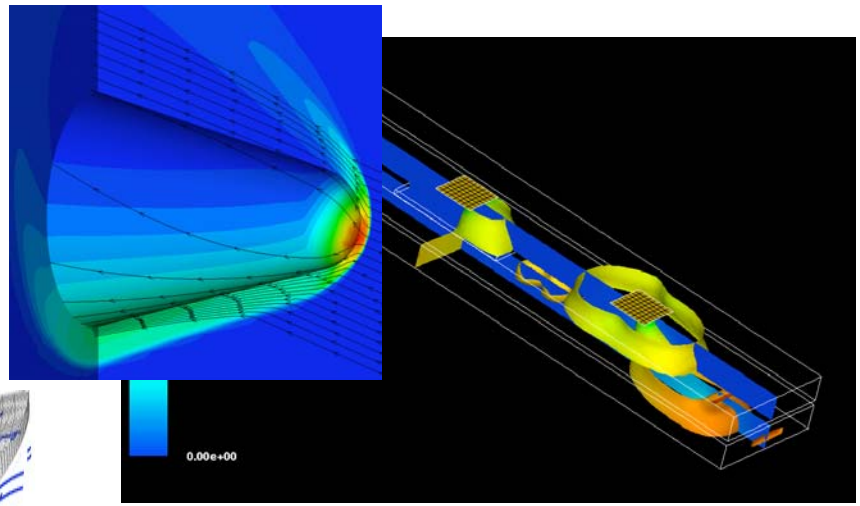
Background/Motivation

Sandia National Laboratories

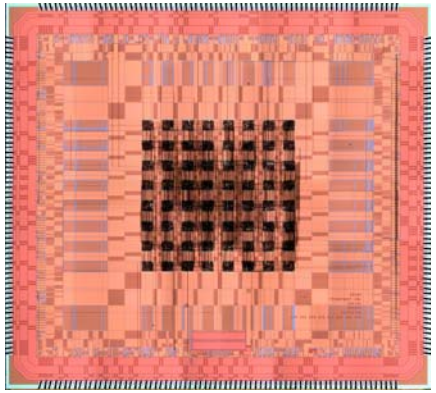
Target Problems: PDES and more…

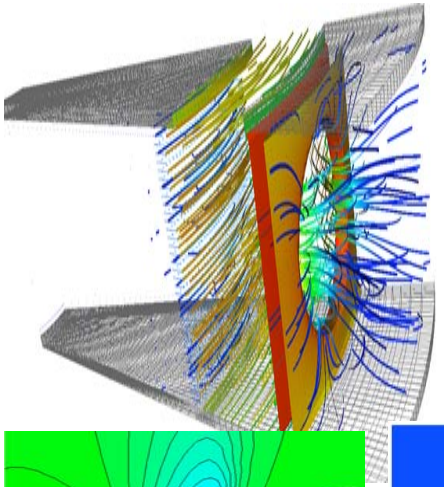


PDES

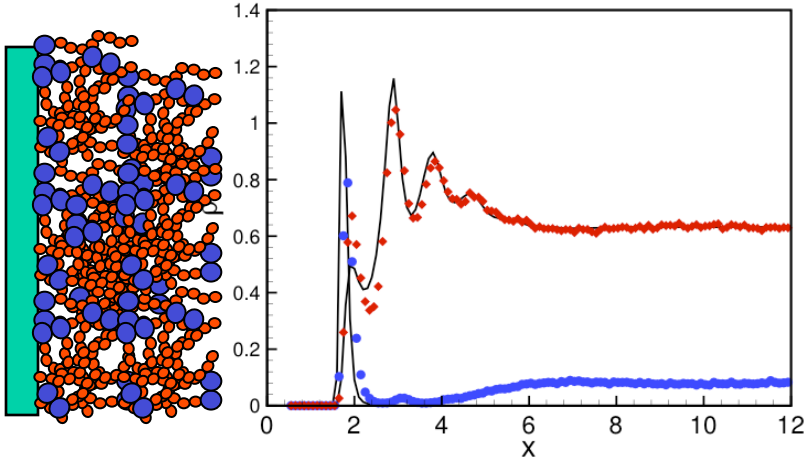


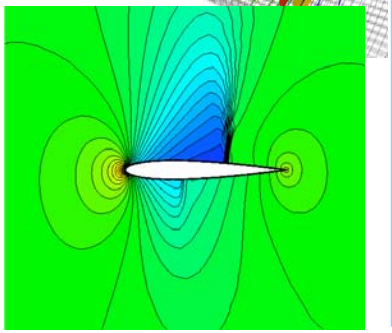
Circuits

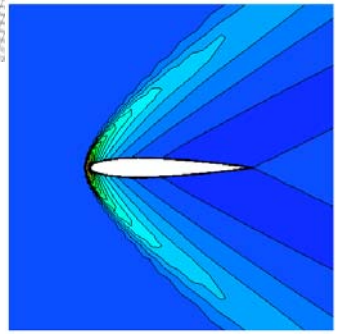




Inhomogeneous Fluids







And More…

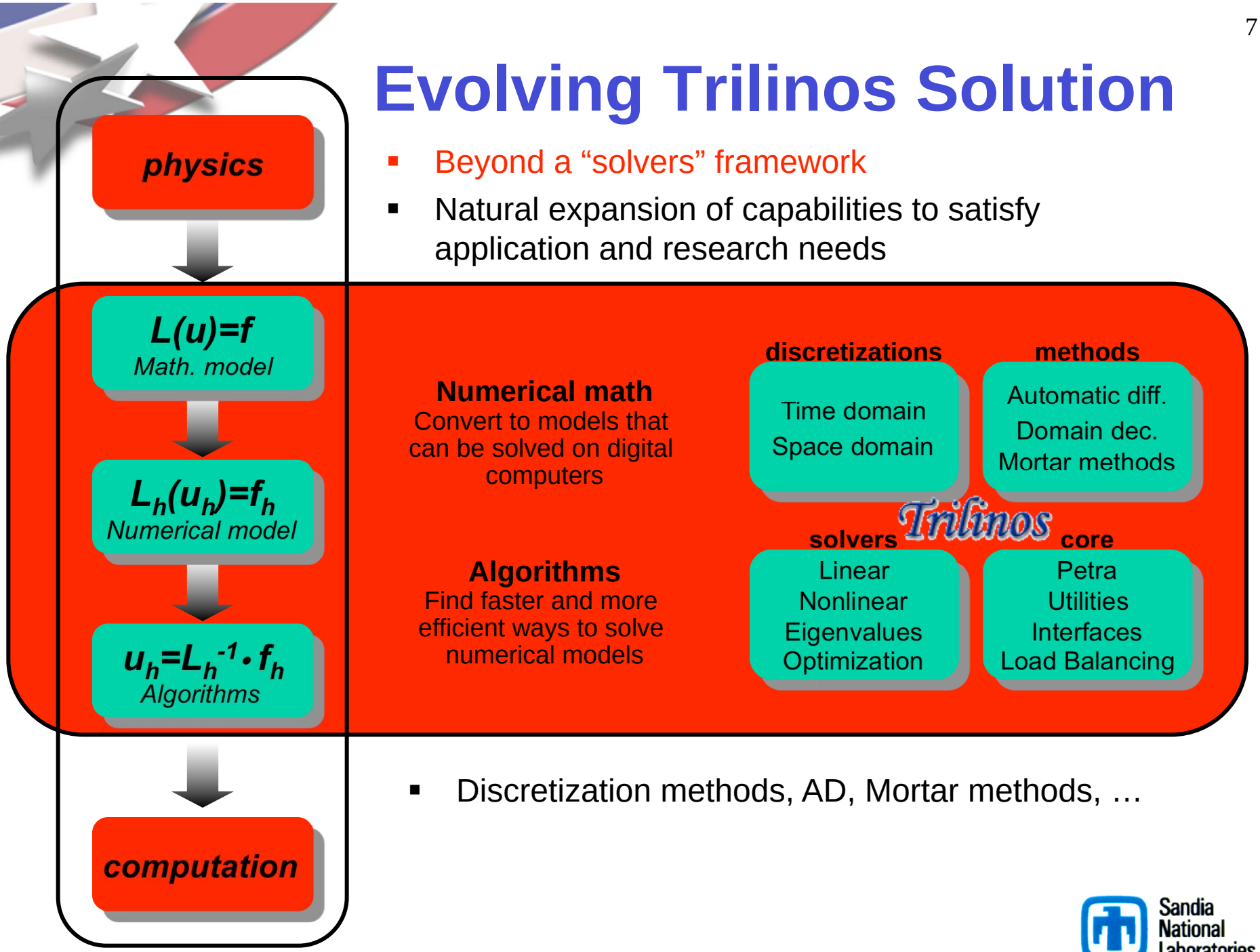
Target Platforms: Any and All
(Now and in the Future)

- Desktop: Development and more…
- Capability machines:
  - Redstorm (XT3), JaguarPF (XT5), Clusters
  - Roadrunner (Cell-based).
  - Multicore nodes.
- Parallel software environments:
  - MPI of course.
  - threads, vectors, CUDA OpenCL, …
  - Combinations of the above.
- User "skins":
  - C++/C, Python
  - Fortran.
  - Web, CCA.

Sandia National Laboratories

Evolving Trilinos Solution

- Beyond a "solvers" framework
- Natural expansion of capabilities to satisfy application and research needs



- Discretization methods, AD, Mortar methods, …

Sandia National Laboratories

Trilinos Package Summary

http://trilinos.sandia.gov

	Objective	Package(s)
Discretizations	Meshing & Spatial Discretizations	phdMesh, Intrepid, Pamgen, Sundance, ITAPS
	Time Integration	Rythmos
Methods	Automatic Differentiation	Sacado
	Mortar Methods	Moertel
Services	Linear algebra objects	Epetra, Jpetra, Tpetra, Kokkos
	Interfaces	Thyra, Stratimikos, RTOp, FEI, Shards
	Load Balancing	Zoltan, Isorropia
	“Skins”	PyTrilinos, WebTrilinos, ForTrilinos, Ctrilinos, Optika
	C++ utilities, I/O, thread API	Teuchos, EpetraExt, Kokkos, Triutils, ThreadPool, Phalanx
Solvers	Iterative (Krylov) linear solvers	AztecOO, Belos, Komplex
	Direct sparse linear solvers	Amesos
	Direct dense linear solvers	Epetra, Teuchos, Pliris
	Iterative eigenvalue solvers	Anasazi, Rbgen
	ILU-type preconditioners	AztecOO, IFPACK, Tifpack
	Multilevel preconditioners	ML, CLAPS
	Block preconditioners	Meros
	Nonlinear system solvers	NOX, LOCA
	Optimization (SAND)	MOOCHO, Aristos, TriKota, Globipack, Optipack
	Stochastic PDEs	Stokhos

Sandia National Laboratories

Capability Leaders:
Layer of Proactive Leadership

- Areas:
  - Framework, Tools & Interfaces (J. Willenbring).
  - Software Engineering Technologies and Integration (R. Bartlett).
  - Discretizations (P. Bochev).
  - Geometry, Meshing & Load Balancing (K. Devine).
  - Scalable Linear Algebra (M. Heroux).
  - Linear & Eigen Solvers (J. Hu).
  - Nonlinear, Transient & Optimization Solvers (A. Salinger).
- Each leader provides strategic direction across all Trilinos packages within area.

Sandia National Laboratories

Package Concepts

Interoperability vs. Dependence

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

- Although most Trilinos packages have no explicit dependence, often packages must interact with *some* other packages:
  - NOX needs operator, vector and linear solver objects.
  - AztecOO needs preconditioner, matrix, operator and vector objects.
  - Interoperability is enabled at configure time. For example, NOX:

    `--enable-nox-lapack` compile NOX lapack interface libraries
    `--enable-nox-epetra` compile NOX epetra interface libraries
    `--enable-nox-petsc` compile NOX petsc interface libraries
- Trilinos `cmake` system is vehicle for:
  - Establishing interoperability of Trilinos components…
  - Without compromising individual package autonomy.

Sandia National Laboratories

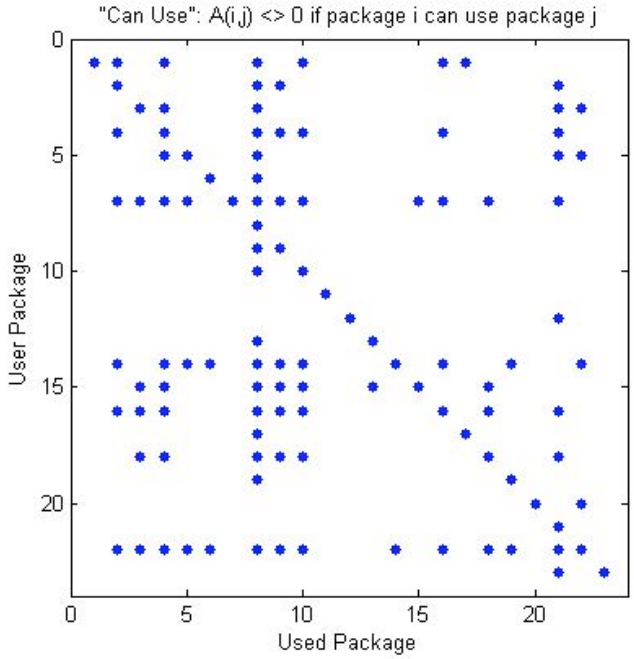
Trilinos Interoperability Mechanisms

(Acquired as Package Matures)

Package builds under Trilinos configure scripts.	➔	*Package* can be built as part of a suite of packages; cross-package interfaces enable/disable automatically
Package accepts user data as Epetra or Thyra objects	➔	Applications using Epetra/Thyra can use *package*
Package accepts parameters from Teuchos ParameterLists	➔	Applications using Teuchos ParameterLists can drive *package*
Package can be used via Thyra abstract solver classes	➔	Applications or other packages using Thyra can use *package*
Package can use Epetra for private data.	➔	*Package* can then use other packages that understand Epetra
Package accesses solver services via Thyra interfaces	➔	*Package* can then use other packages that implement Thyra interfaces
Package available via PyTrilinos	➔	*Package* can be used with other Trilinos packages via Python.

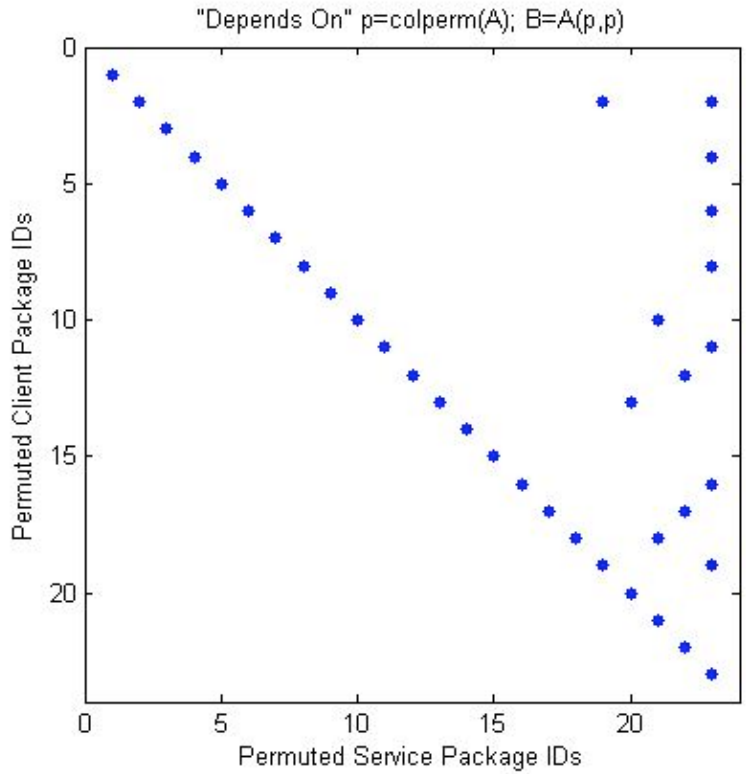
Sandia National Laboratories

“Can Use” vs. “Depends On”



“Can Use”

- Interoperable without dependence.
- Dense is Good.
- Encouraged.



“Depends On”

- OK, if essential.
- Epetra, Teuchos: 9 clients.
- Thyra, NOX: 2 clients.
- Discouraged.

What Trilinos is not …

- Trilinos is not a single monolithic piece of software. Each package:
  - Can be built independent of Trilinos.
  - Has its own self-contained CVS structure.
  - Has its own Bugzilla product and mail lists.
  - Development team is free to make its own decisions about algorithms, coding style, release contents, testing process, etc.
- Trilinos top layer is not a large amount of source code: ~1.5%
- Trilinos is not “indivisible”:
  - You don’t need all of Trilinos to get things done.
  - Any collection of packages can be combined and distributed.
  - Upcoming public release contains ~45 of the 50+ Trilinos packages.

Whirlwind Tour of Packages

Core Utilities

Discretizations Methods Solvers

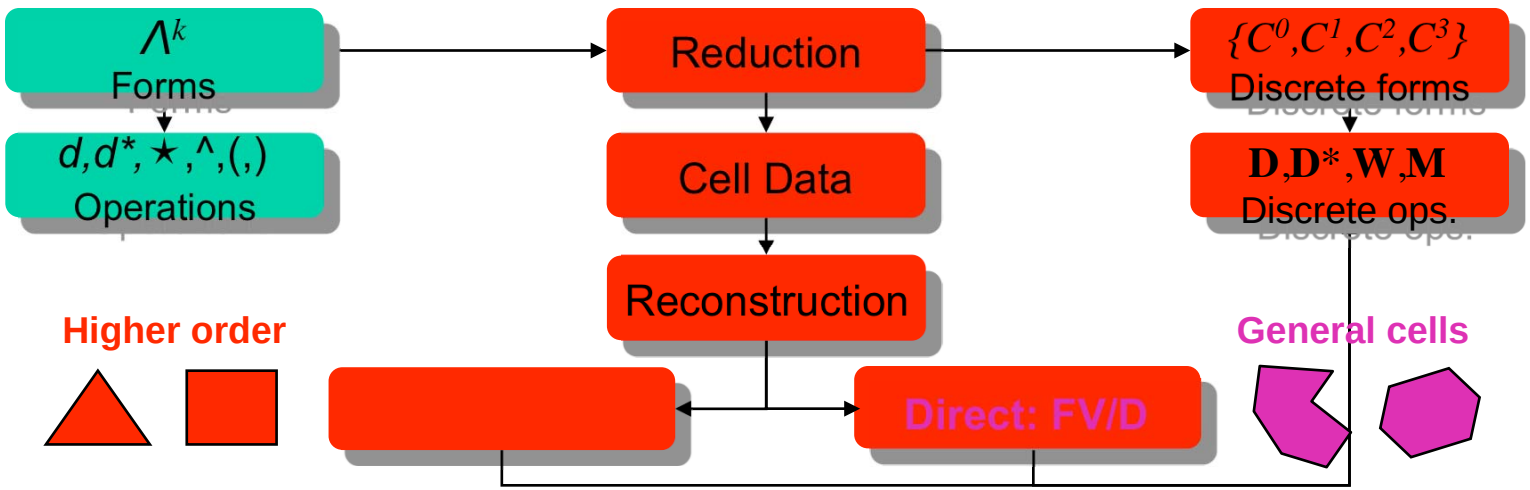
Sandia National Laboratories

Intrepid

Interoperable Tools for Rapid Development of Compatible Discretizations

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

- allows access to FEM, FV and FD 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
- Includes: backward Euler, forward Euler, explicit Runge-Kutta, and implicit BDF at this time.
- Native support for operator split methods.
- Highly modular.
- Forward sensitivity computations will be included in the first release with adjoint sensitivities coming in near future.

Developers: Todd Coffey, Roscoe Bartlett

Whirlwind Tour of Packages

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

Sacado: Automatic Differentiation

- Efficient OO based AD tools optimized for element-level computations

- Applies AD at "element"-level computation
  - "Element" means finite element, finite volume, network device,…

- Template application's element-computation code
  - Developers only need to maintain one templated code base

- 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, $\partial f / \partial x$ 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), $\partial f / \partial x$ 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**

Sandia National Laboratories

Teuchos

- Portable utility package of commonly useful tools:
  - ParameterList class: key/value pair database, recursive capabilities.
  - LAPACK, BLAS wrappers (templated on ordinal and scalar type).
  - Dense matrix and vector classes (compatible with BLAS/LAPACK).
  - FLOP counters, timers.
  - Ordinal, Scalar Traits support: Definition of ‘zero’, ‘one’, etc.
  - Reference counted pointers / arrays, and more…
- Takes advantage of advanced features of C++:
  - Templates
  - Standard Template Library (STL)
- Teuchos::ParameterList:
  - Allows easy control of solver parameters.
  - XML format input/output.

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

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 3 implementations under development.

[1]Petra is Greek for “foundation”.

Petra Implementations

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

- Tpetra (Templated Petra):
  - Next generation C++ version.
  - Templated scalar and ordinal fields.
  - Uses namespaces, and STL: Improved usability/efficiency.
- Jpetra (Java Petra):
  - Pure Java. Portable to any JVM.
  - Interfaces to Java versions of MPI, LAPACK and BLAS via interfaces.

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

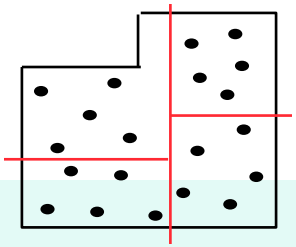
Sandia National Laboratories

EpetraExt: Extensions to Epetra

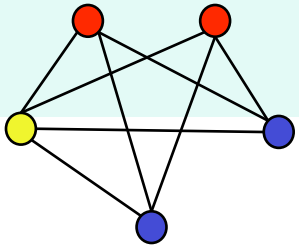
- Library of useful classes not needed by everyone
- Most classes are types of “transforms”.
- Examples:
  - Graph/matrix view extraction.
  - Epetra/Zoltan interface.
  - Explicit sparse transpose.
  - Singleton removal filter, static condensation filter.
  - Overlapped graph constructor, graph colorings.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

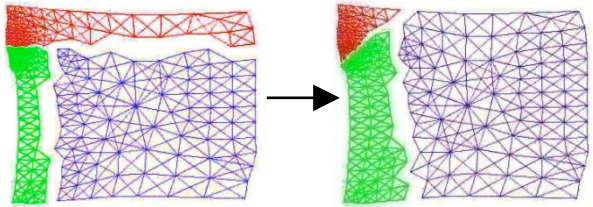
Developer: Robert Hoekstra, Alan Williams, Mike Heroux

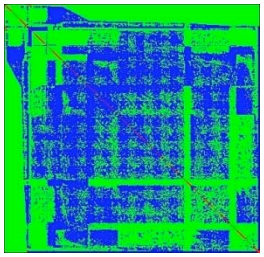
Zoltan



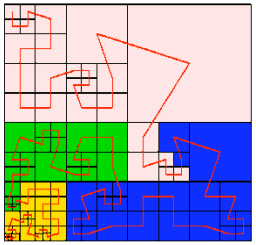
- Data Services for Dynamic Applications
  - Dynamic load balancing
  - Graph coloring
  - Data migration
  - Matrix ordering



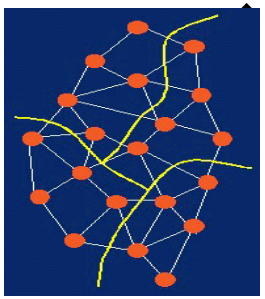




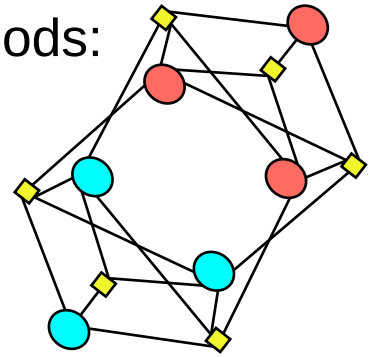
- Partitioners:



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)



Developers: Karen Devine, Eric Boman, Robert Heaphy

Sandia National Laboratories

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

Sandia National Laboratories

"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, and Marzio Sala

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

Sandia National Laboratories

AztecOO

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

- Aztec is the workhorse solver at Sandia:
  - 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 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.

Developers: Mike Heroux, Alan Williams, Ray Tuminaro

Sandia National Laboratories

Belos

- Next-generation linear solver library, written in templated C++.

- Provide a generic framework for developing iterative algorithms for solving large-scale, linear problems.
- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - **Operator-vector products:** `Belos::MultiVecTraits, Belos::OperatorTraits`
  - **Orthogonalization:** `Belos::OrthoManager, Belos::MatOrthoManager`
  - **Status tests:** `Belos::StatusTest, Belos::StatusTestResNorm`
  - **Iteration kernels:** `Belos::Iteration`
  - **Linear solver managers:** `Belos::SolverManager`
- AztecOO provides solvers for $Ax=b$; what about solvers for:
  - Simultaneously solved systems w/ multiple-RHS: $AX = B$
  - Sequentially solved systems w/ multiple-RHS: $AX_i = B_i,\ i=1,\ldots,t$
  - Sequences of multiple-RHS systems: $A_iX_i = B_i,\ i=1,\ldots,t$
- Many advanced methods for these types of linear systems
  - Block methods: block GMRES [Vital], block CG/BICG [O'Leary]
  - "Seed" solvers: hybrid GMRES [Nachtigal, et al.]
  - Recycling solvers: recycled Krylov methods [Parks, et al.]
  - Restarting techniques, orthogonalization techniques, …

Developers: Heidi Thornquist, Mike Heroux, Mike Parks, Rich Lehoucq, Teri Barth

Sandia National Laboratories

IFPACK: Algebraic Preconditioners

- Overlapping Schwarz preconditioners with incomplete factorizations, block relaxations, block direct solves.

- Accept user matrix via abstract matrix interface (Epetra versions).
- Uses Epetra for basic matrix/vector calculations.
- Supports simple perturbation stabilizations and condition estimation.
- Separates graph construction from factorization, improves performance substantially.
- Compatible with AztecOO, ML, Amesos. Can be used by NOX and ML.

Developers: Marzio Sala, Mike Heroux

Sandia National Laboratories

: Multi-level Preconditioners

- Smoothed aggregation, multigrid and domain decomposition preconditioning package

- Critical technology for scalable performance of some key 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 completely independent of other Trilinos packages.

Developers: Ray Tuminaro, Jonathan Hu, Marzio Sala

Anasazi

- Next-generation eigensolver library, written in templated C++.

- Provide a generic framework for developing iterative algorithms for solving large-scale eigenproblems.
- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - **Operator-vector products:** `Anasazi::MultiVecTraits`, `Anasazi::OperatorTraits`
  - **Orthogonalization:** `Anasazi::OrthoManager`, `Anasazi::MatOrthoManager`
  - **Status tests:** `Anasazi::StatusTest`, `Anasazi::StatusTestResNorm`
  - **Iteration kernels:** `Anasazi::Eigensolver`
  - **Eigensolver managers:** `Anasazi::SolverManager`
  - **Eigenproblem:** `Anasazi::Eigenproblem`
  - **Sort managers:** `Anasazi::SortManager`
- Currently has solver managers for three eigensolvers:
  - Block Krylov-Schur
  - Block Davidson
  - LOBPCG
- Can solve:
  - standard and generalized eigenproblems
  - Hermitian and non-Hermitian eigenproblems
  - real or complex-valued eigenproblems

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

Sandia National Laboratories

NOX: Nonlinear Solvers

- Suite of nonlinear solution methods

Broyden's Method

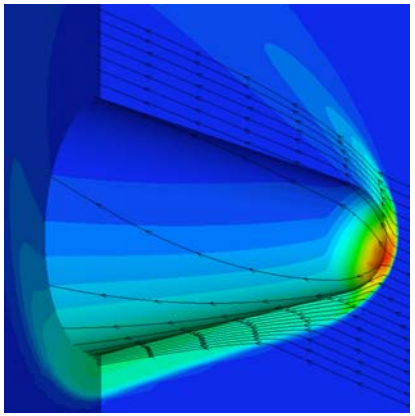
$$M_B = f(x_c) + B_c d$$

Newton's Method

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

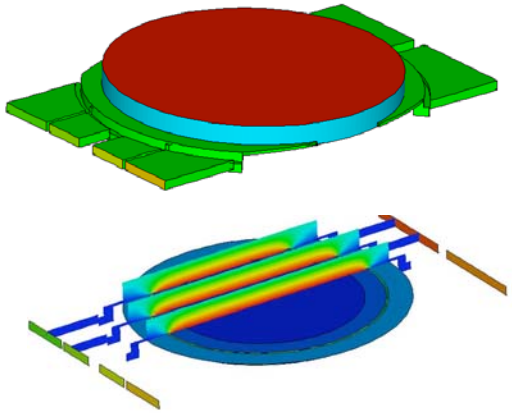
Tensor Method

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



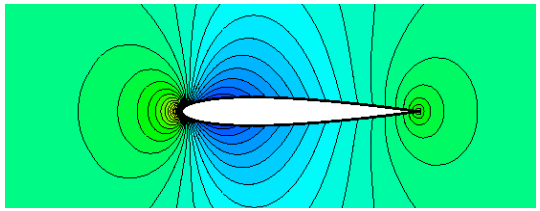
Globalizations

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 system solves.

Developer: Denis Ridzal

Sandia National Laboratories

Full Vertical Solver Coverage

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**

Solver Collaborations: ANAs, LALs and APPs

Sandia National Laboratories

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

Sandia National Laboratories

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

for $i = 1, 2, \ldots$

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

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

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

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

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

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

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

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

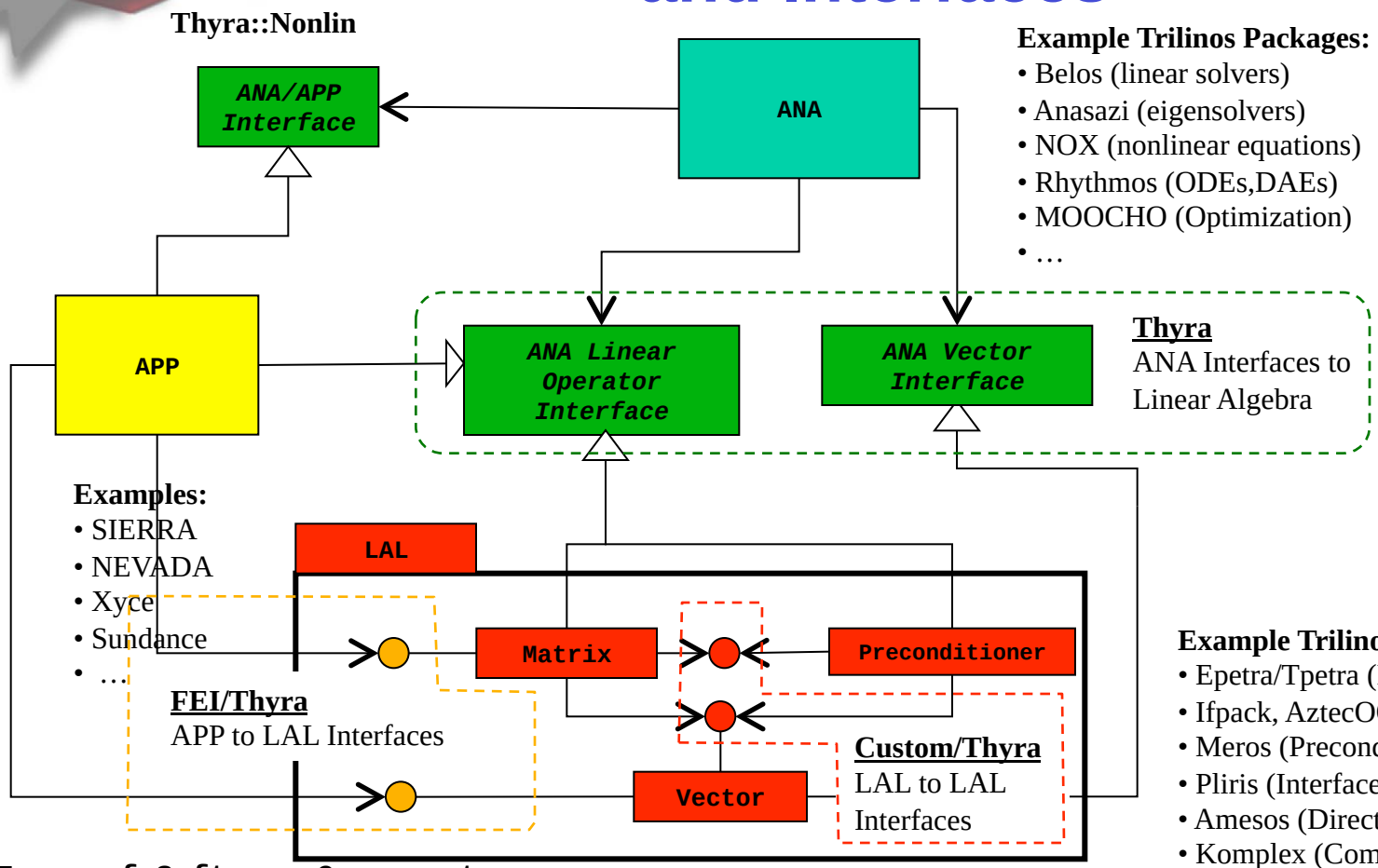
check convergence; continue if necessary

end

Types of operations	Types of objects
linear operator applications	Linear Operators • A
vector-vector operations	Vectors • r, x, p, q
scalar operations	Scalars • ρ, β, γ, α
scalar product <x,y> defined by vector space	Vector spaces? • $\mathcal{X}$

Sandia National Laboratories

Solver Software Components and Interfaces



Example Trilinos Packages:
- Belos (linear solvers)
- Anasazi (eigensolvers)
- NOX (nonlinear equations)
- Rhythmos (ODEs,DAEs)
- MOOCHO (Optimization)
- …

Examples:
- SIERRA
- NEVADA
- Xyce
- Sundance
- …

Example Trilinos Packages:
- Epetra/Tpetra (Mat,Vec)
- Ifpack, AztecOO, ML (Preconditioners)
- Meros (Preconditioners)
- Pliris (Interface to direct solvers)
- Amesos (Direct solvers)
- Komplex (Complex/Real forms)
- …

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

- Stratimikos created Greek words "stratigiki“ (strategy) and "grammikos“ (linear)
- Defines class Thyra::DefaultLinearSolverBuilder.
- Provides common access to:
    - Linear Solvers: Amesos, AztecOO, Belos, …
    - Preconditioners: Ifpack, ML, …
  - Reads in options through a parameter list (read from XML?)
  - Accepts any linear system objects that provide
    - Epetra_Operator / Epetra_RowMatrix view of the matrix
    - SPMD vector views for the RHS and LHS (e.g. Epetra_[Multi]Vector objects)
- Provides uniform access to linear solver options that can be leveraged across multiple applications and algorithms

Key Points

- Stratimikos is an important building block for creating more sophisticated linear solver capabilities!

Sandia National Laboratories

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

Trilinos Integration into an Application

Where to start?

http://trilinos.sandia.gov

Export Makefile System

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

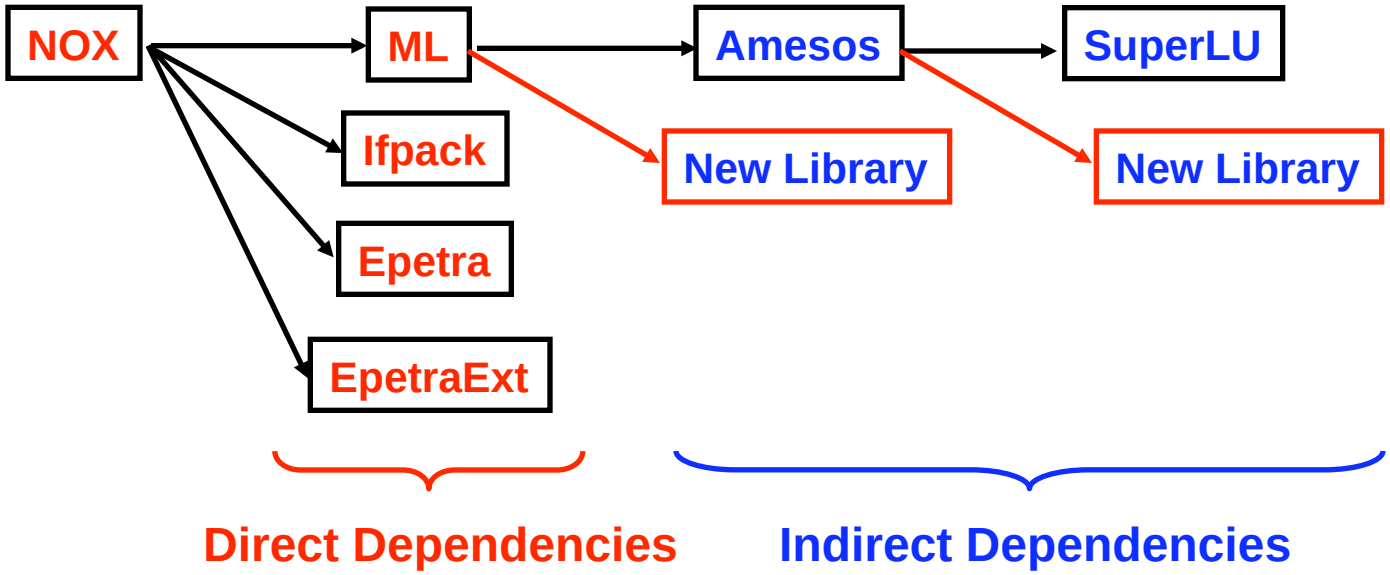
There are a number of issues:

- **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

- The number of packages in Trilinos has exploded.
- 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.

Sandia National Laboratories

Export Makefiles in Action

```
#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 search paths
# for libraries and headers
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
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) $<
```

Sandia National Laboratories

Concluding Remarks

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

Sandia National Laboratories

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/

Sandia National Laboratories

Trilinos Availability / Information

- Trilinos and related packages are available via LGPL.
- Current release (10.0) is “click release”. Unlimited availability.
- Trilinos Release 10.1: January 2010.
- Trilinos Awards:
  - 2004 R&D 100 Award.
  - SC2004 HPC Software Challenge Award.
  - Sandia Team Employee Recognition Award.
  - Lockheed-Martin Nova Award Nominee.
- More information:
  - http://trilinos.sandia.gov
- Annual Forums:
  - DOE ACTS Tutorial (3rd week in August).
  - Annual Trilinos User Group Meeting in November @ SNL
    - talks available for download

Sandia National Laboratories

Useful Links

Trilinos website: http://trilinos.sandia.gov

Trilinos tutorial: http://trilinos.sandia.gov/Trilinos10.0Tutorial.pdf

Trilinos mailing lists: http://trilinos.sandia.gov/mail_lists.html

Trilinos User Group (TUG) meetings:

http://trilinos.sandia.gov/events/trilinos_user_group_2008

http://trilinos.sandia.gov/events/trilinos_user_group_2007

Trilinos Hands-On Tutorial

http://code.google.com/p/trilinos

Teuchos Package

- For many Trilinos packages, this is the only required or “depends on” package.
- Provides basic utilities:
  - Parameter List
  - Memory management/Smart Pointer classes
  - Command Line Parser
  - Templated BLAS/LAPACK interfaces
  - XML Parser
  - MPI Communicator

Sandia National Laboratories

Parameter List

- A key/value pair database that is recursive
  - Uses an implementation of the boost::Any object
  - Can read or output to XML files (internal xml or link to external xml)
  - Recursive: Sublists – nesting of parameter lists within itself
- Primary means of setting parameters in Trilinos packages:

```
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 max_iters = p.get<int>("Max Iterations");
int fill = p.sublist("Solver Options").get<int>("Fill Factor");
```

Sandia National Laboratories

Reference Counted Smart Pointer

- Powerful memory management for Trilinos packages!
- A wrapper for a pointer so that you don't have to explicity deallocate the memory.
  - When last RCP to the object is deleted, the underlying memory is deallocated.
- Next C++ standard will have Boost Smart Pointers

```
class A {

};

int main {

  A* a  = new A;
        .
        .
        .
  delete a;
}
```

```
class A {

};

int main {

  A* a  = new A;

  using namespace Teuchos;

  RCP<A> a = rcp(new A);
  RCP<A> b = a;
}
```

Sandia National Laboratories

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.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

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

Trilinos/doc/RCPbeginnersGuide

Sandia National Laboratories

Time Monitor

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

```
RCP<Time> fill_timer = TimeMonitor::getNewTimer("Fill Time");
```

- When TimeMonitor is created, the timer starts:

```
TimeMonitor tm(Time& t);
```

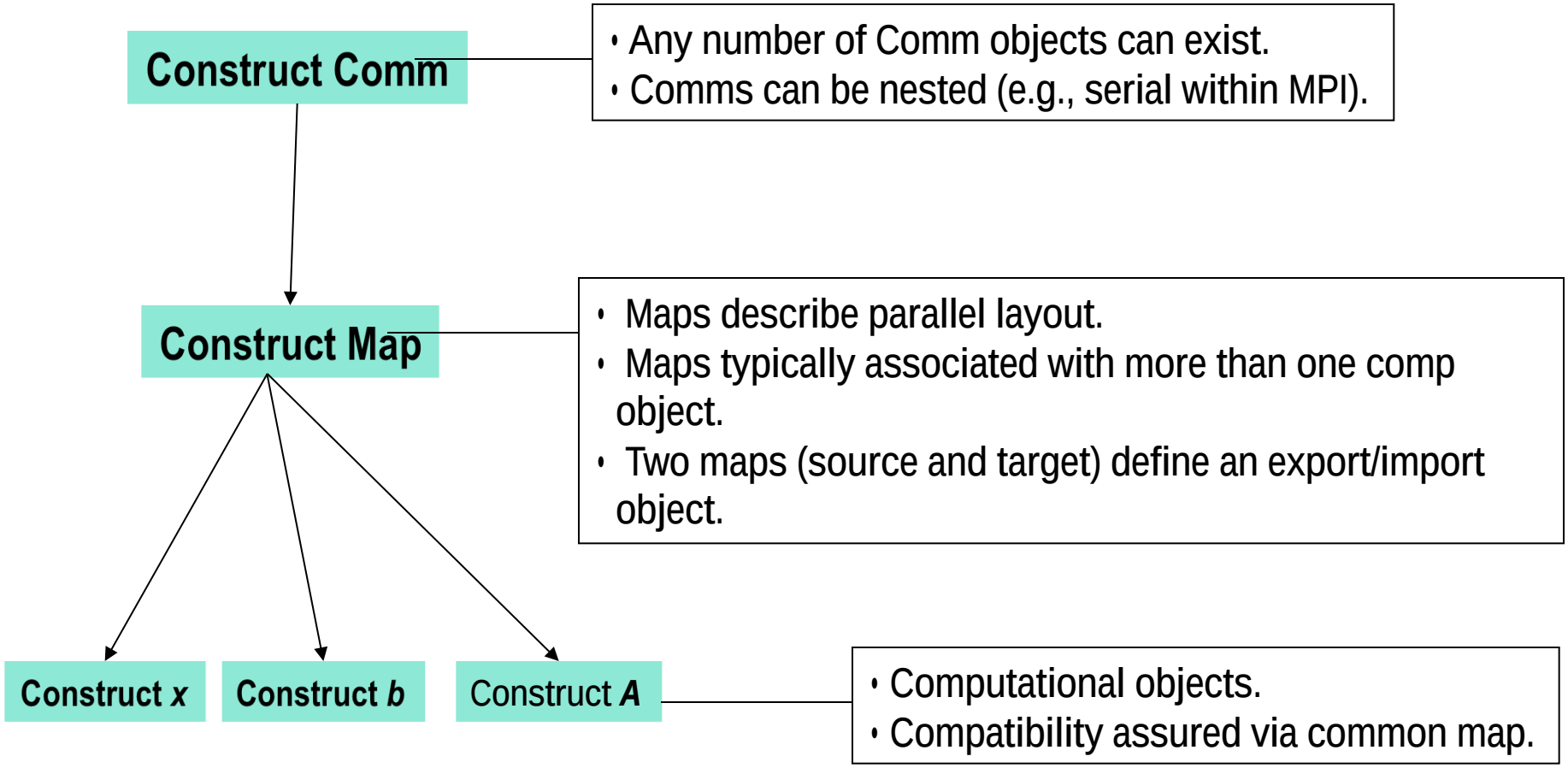
- When TimeMonitor is destroyed (usually when you leave scope), the timer stops.

Epetra Package

Linear Algebra Package

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

Typical Flow of Epetra Object Construction



Sandia National Laboratories

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;
}
```

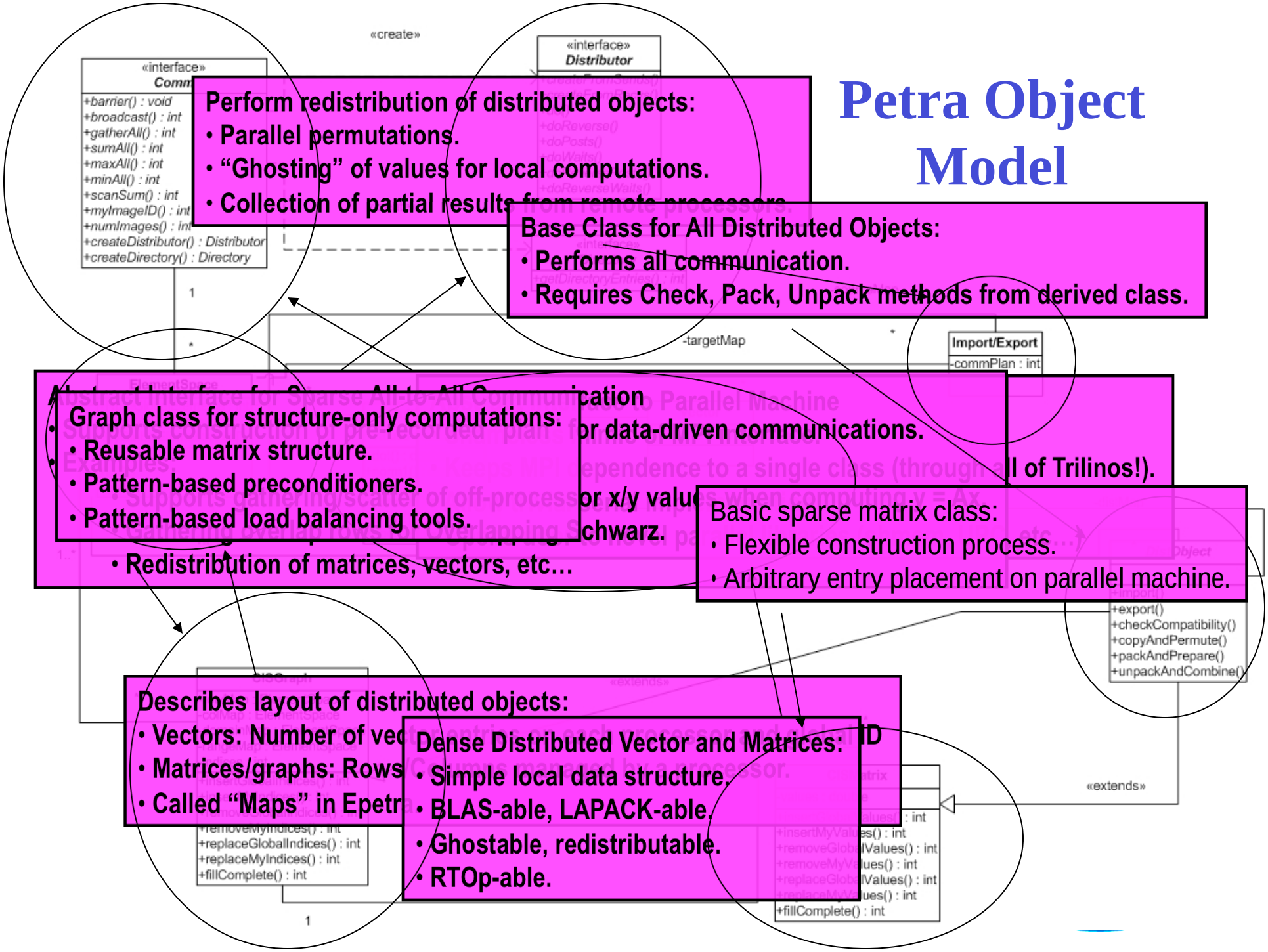
Sandia National Laboratories

Petra Implementations

- Three version under development:
- **Epetra** (Essential Petra):
  - Current production version.
  - Restricted to real, double precision arithmetic.
  - Uses stable core subset of C++ (circa 2000).
  - Interfaces accessible to C and Fortran users.
- **Tpetra** (Templated Petra):
  - Next generation C++ version.
  - Templated scalar and ordinal fields.
  - Uses namespaces, and STL: Improved usability/efficiency.
  - Advanced node architecture, multiprecision support.
- **Jpetra** (Java Petra):
  - Pure Java. Portable to any JVM.
  - Interfaces to Java versions of MPI, LAPACK and BLAS via interfaces.

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.

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!**

Sandia National Laboratories

1-to-1 Maps

- *1-to-1 map* (defn): A map is 1-to-1 if each GID appears only once in the map (and is therefore associated with only a single processor).
- Certain operations in parallel data repartitioning require 1-to-1 maps. Specifically:
  - The source map of an import must be 1-to-1.
  - The target map of an export must be 1-to-1.
  - The domain map of a 2D object must be 1-to-1.
  - The range map of a 2D object must be 1-to-1.

Sandia National Laboratories

2D Objects: Four Maps

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

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

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

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

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

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

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

Original Problem

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

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

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

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?

- A bunch of stuff.
- One task is to create (if needed) import/export objects to support distributed matrix-vector multiplication:
  - If ColMap ≠ DomainMap, create Import object.
  - If RowMap ≠ RangeMap, create Export object.
- A few rules:
  - Rectangular matrices will *always* require:

    A.FillComplete(DomainMap,RangeMap);
  - DomainMap and RangeMap *must be 1-to-1*.

Sandia National Laboratories

Linear System Solves

Sandia National Laboratories

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.

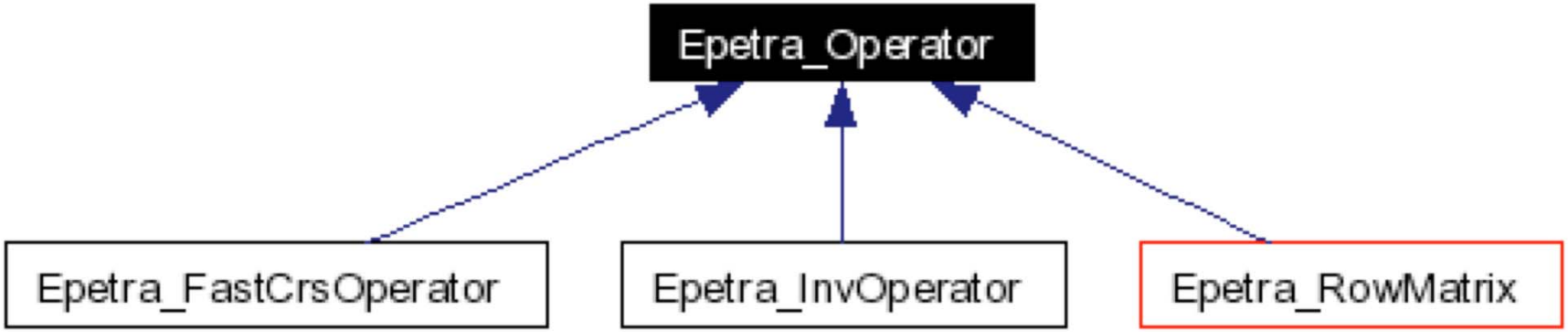
Sandia National Laboratories

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.

Sandia National Laboratories

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).

Sandia National Laboratories

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)

Sandia National Laboratories

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.

Anasazi / Belos Design

- Eigenproblem/ LinearProblem Class
  - Describes the problem and stores the answer
- Eigensolver / Linear Solver Manager (SolverManager) Class
  - Parameter list driven strategy object describing behavior of solver
- Eigensolver / Iteration Class
  - Provide basic iteration interface.
- MultiVecTraits and OperatorTraits
  - Traits classes for interfacing linear algebra
- SortManagerClass [Anasazi only]
  - Allows selection of desired eigenvalues
- OrthoManagerClass
  - Provide basic interface for orthogonalization
- StatusTestClass
  - Control testing of convergence, etc.
- OutputManagerClass
  - Control verbosity and printing in a MP scenario

Sandia National Laboratories

Anasazi / Belos Status

- Anasazi (Trilinos Release 8.0):
  - Solvers: Block Krylov-Schur, Block Davidson, LOBPCG
  - Can solve standard and generalized eigenproblems
  - Can solve Hermitian and non-Hermitian eigenproblems
  - Can target largest or smallest eigenvalues
  - Block size is independent of number of requested eigenvalues
- Belos (Trilinos Release 8.0):
  - Solvers: CG, BlockCG, BlockGMRES, BlockFGMRES, GCRO-DR
  - Belos::EpetraOperator, Thyra::LOWS, and Stratimikos interface allows for integration into other codes
  - Block size is independent of number of right-hand sides
- Linear algebra adapters for Epetra, NOX/LOCA, and Thyra
- Epetra interface accepts Epetra_Operators, so can be used with ML, AztecOO, Ifpack, Belos, etc…
- Configurable via Teuchos::ParameterList

Sandia National Laboratories

Preconditioning

Sandia National Laboratories

: AMG for Magnetics Simulations

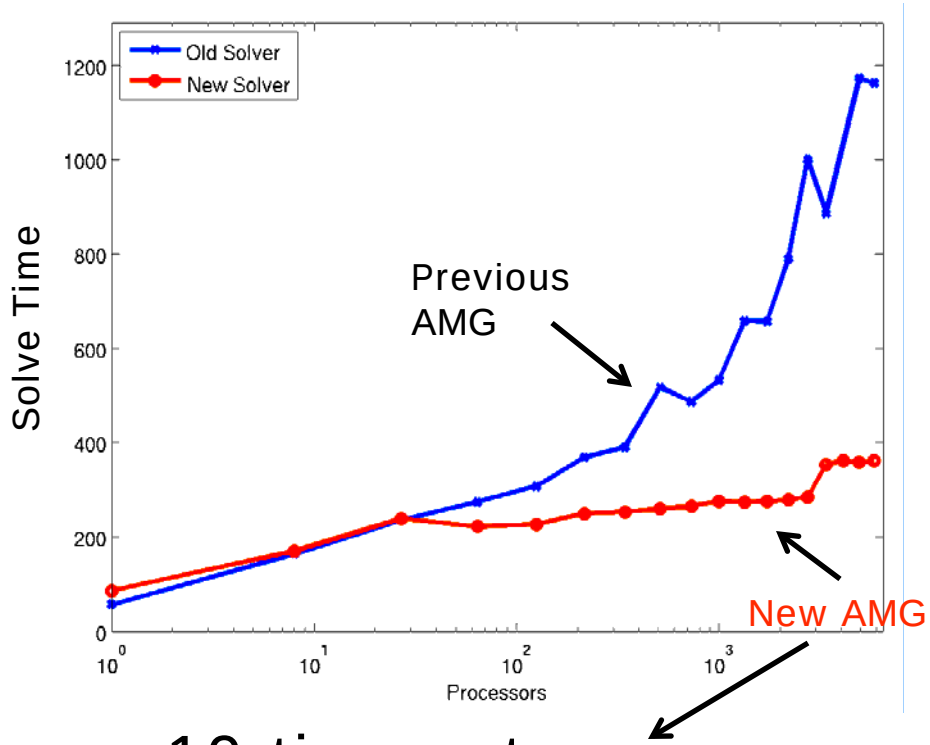
$$\nabla \times \frac{1}{\mu} \nabla \times \mathrm{E} + \sigma \, \mathrm{E} = 0 \quad \text{in } \Omega \qquad (*)$$

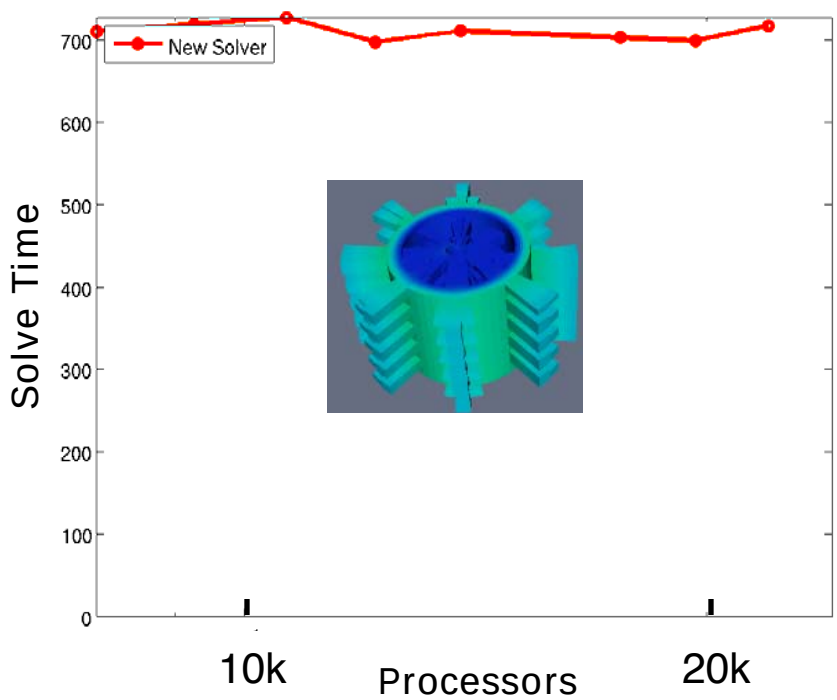
- Efficient solution critical to HEDP Z-pinch simulations
- <u>Challenges:</u>
  - Standard solvers do not converge
  - Large near null space of curl
  - Conductivity variation
  - Mesh stretching
- Two Sandia AMG methods for eddy current eqns:
  - (2002) Specialized AMG satisfying commutating relationship
  - (2006) Implicitly reformulate (*) and leverage ML standard AMG:

$$\begin{bmatrix} \Delta^{(e)} & \nabla \\ \nabla \cdot & \Delta^{(n)} \end{bmatrix}$$

Sandia National Laboratories

: Scaling on Red Storm





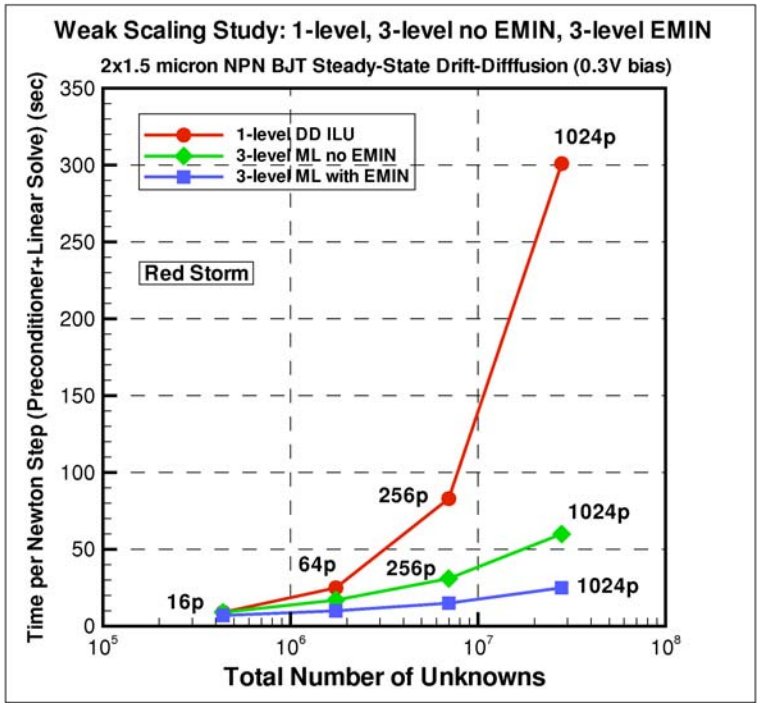
- 10 times steps
- Chebyshev polynomial smoother
- Parallel load-balancing via Zoltan

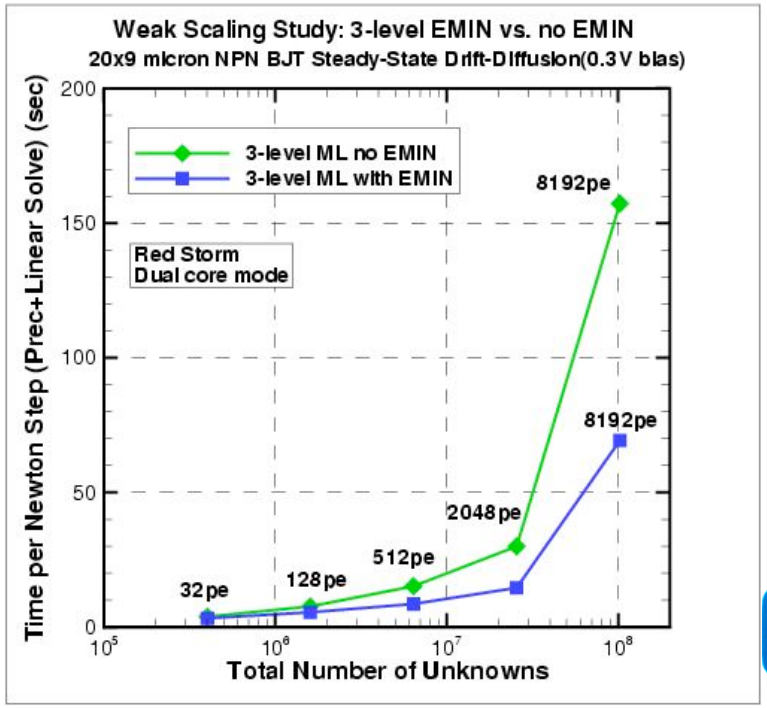
Sandia National Laboratories

: AMG for non-symmetric systems

- For many applications, $A \neq A^T$
  - Symmetric AMG methods ineffective or <u>diverge</u>
  - Non-symmetric AMG theory doesn't exist
- ML has new AMG method for $A \neq A^T$ that minimizes basis function energy (Tuminaro & Sala)

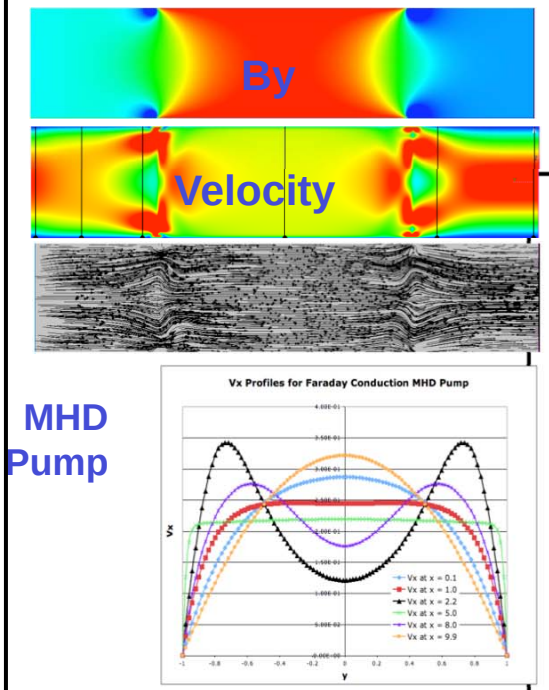
Charon drift-diffusion (P. Lin, J. Shadid, et al.)

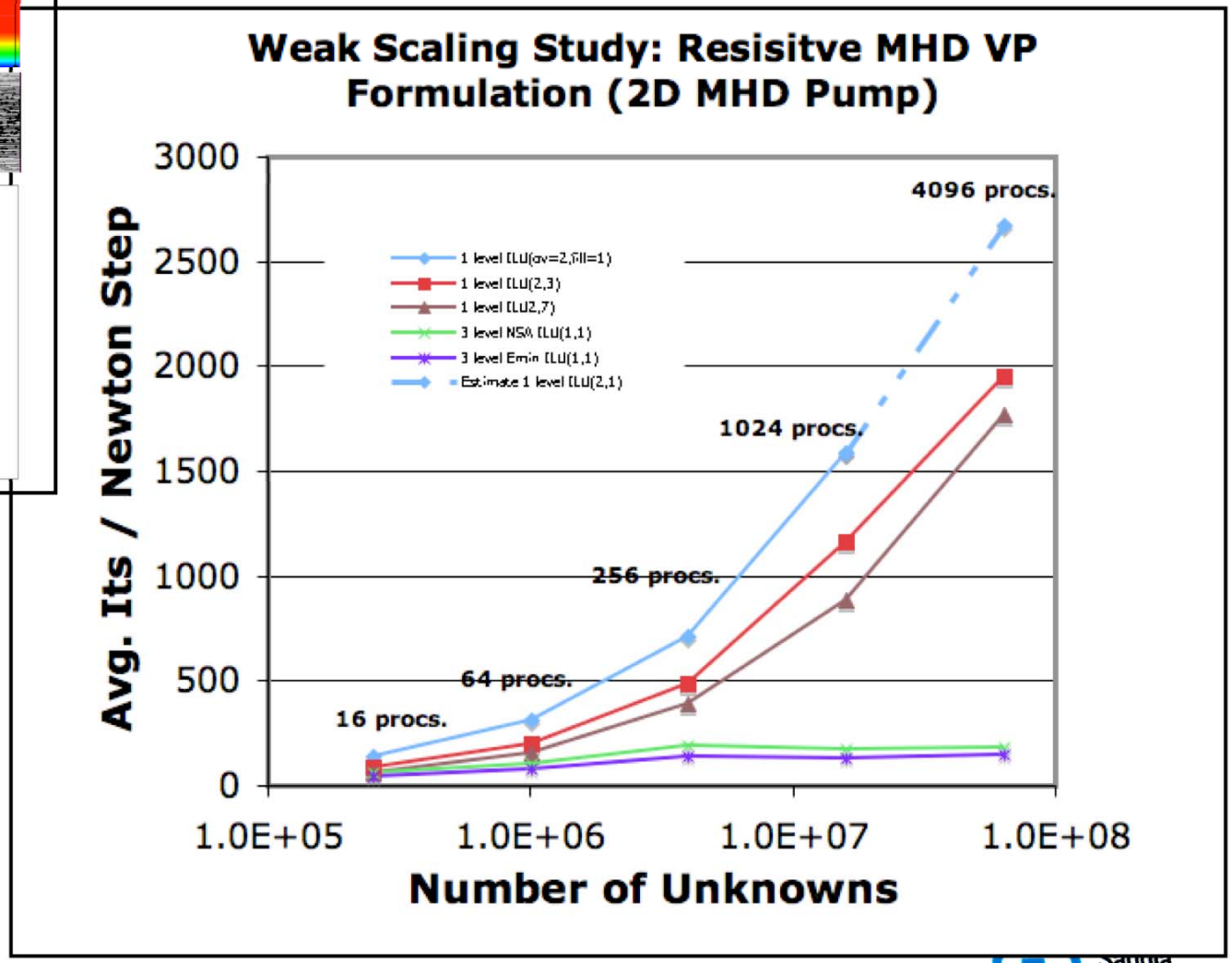




Sandia National Laboratories

Scalability
(MHD Pump, Cray XT3)





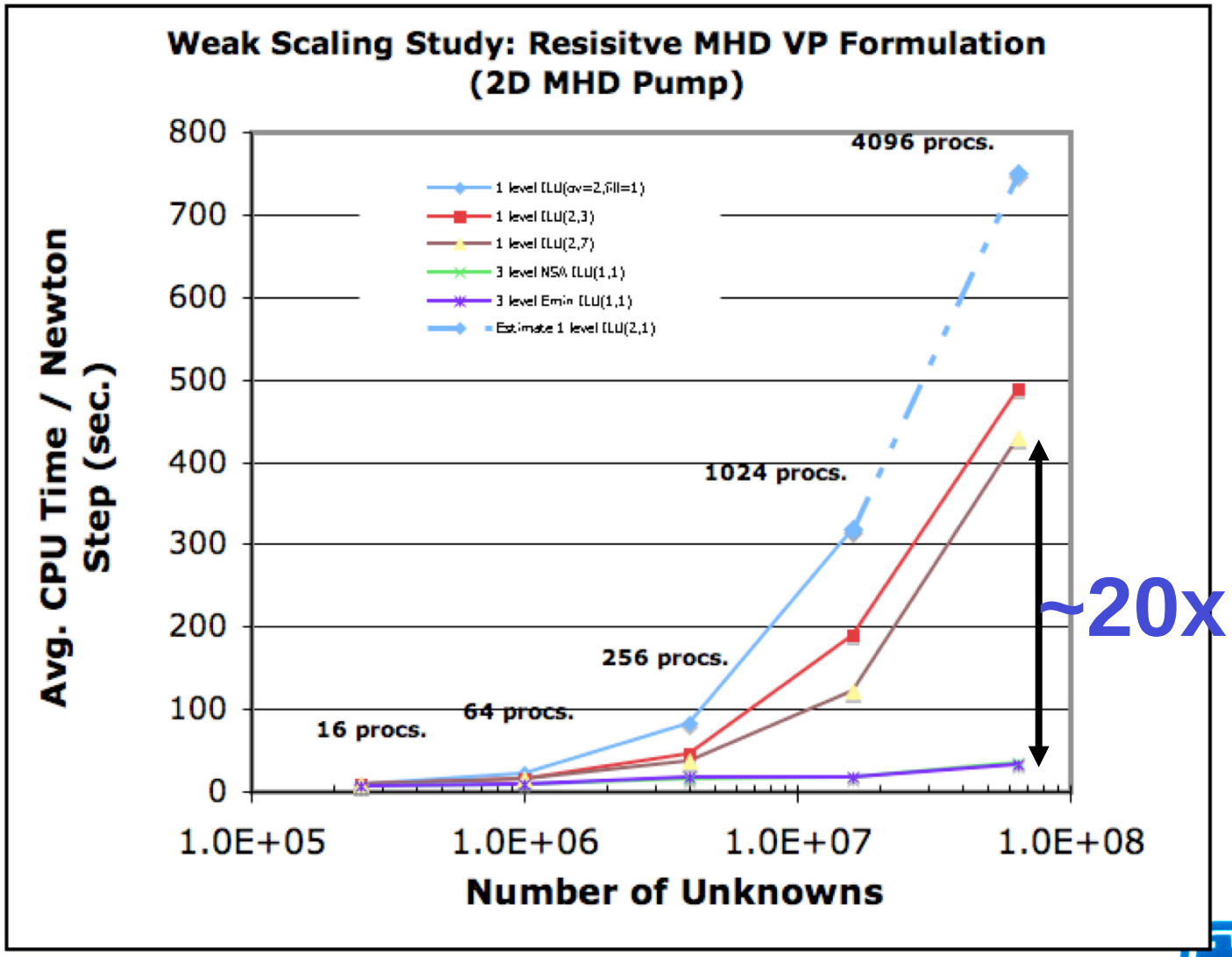
Preconditioners
- 1-level ILU(2,1)
- 1-level ILU(2,3)
- 1-level ILU(2,7)
- 3-level ML(NSA,Gal)
- 3-level ML(EMIN, PG)

ML: Tuminaro, Hu
Ifpack: Heroux

Sandia National Laboratories

Scalability

(MHD Pump, Cray XT3)



Sandia National Laboratories

Multicore
(Inter-core comm. with MPI)

Nodes	Cores per node	Compute Jac+Prec		Linear Solve		Total	
		Time (sec)	Eff.	Time (sec)	Eff.	Time (sec)	Eff.
4096	1	16.9	-------	4.3	-------	21.2	------
2048	2	18.2	93%	4.5	95%	22.6	94%
1024	4	17.7	95%	4.9	88%	22.6	94%

Multi-core Efficiency Study
New 2.2 GHz Quad Cores
Cray XT3/4 (09/29/08)
Total of 4096 cores

12800x1280 mesh: ~65M unknowns; Agg = 33;
Coarse Operator: ~60K unknowns
ML: V(1,1) with ILU(1,2)/ ILU(1,2)/KLU and Petrov-Galerkin Projection

<u>Our Largest Steady-state Simulation to Date:</u>
1+ Billion unknowns
250 Million Quad elements
24,000 cores Cray XT3/4
Newton-GMRES / ML: PG-AMG 4 level
18 Newton steps
86 Avg. No. Linear Its. / Newton step
33 min. for solution

Sandia National Laboratories

Nonlinear System Solves

Sandia National Laboratories

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

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

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

Sandia National Laboratories

Building Blocks of NOX

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

Iterate Control (Solver)

- Choose an initial guess x_0
- For $k = 0,1,2,\ldots$
  - Calculating the Direction
    - Compute $F_k = F(x_k)$
    - Compute J_k where $(J_k)_{ij} = \partial F_i(x_k)/\partial x_j$
    - Let $d_k = -J_k^{-1} F_k$
  - Damping or Line Search
    - (Optional) Let l_k be a calculated step length
    - Set $x_{k+1} = x_k + l_k d_k$
  - Stopping Criteria (Status Test)
    - Test for Convergence or Failure

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 DX: {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.

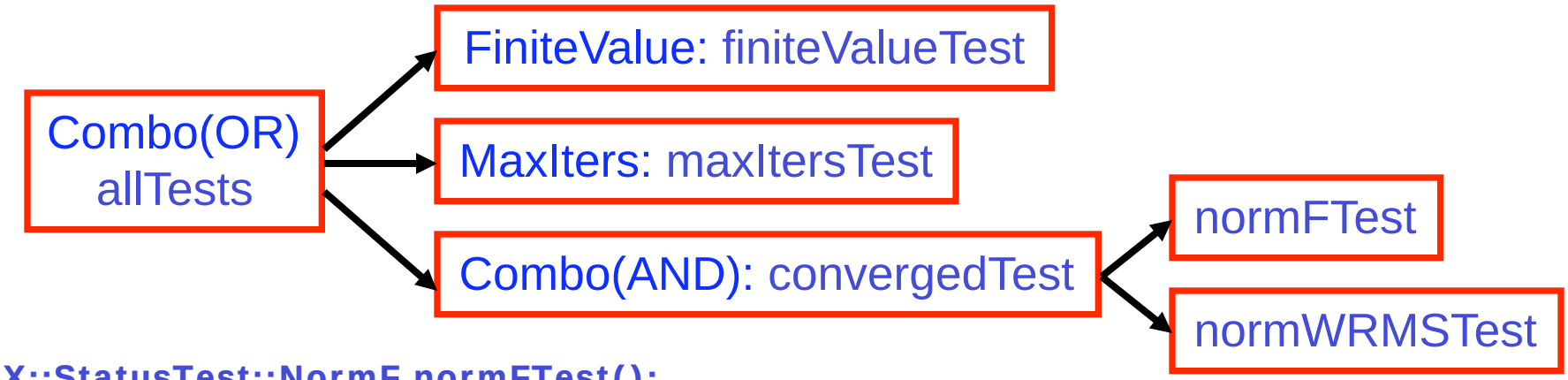
$$\frac{\|F_k\|}{\|F_{k-1}\|} \geq \text{tol}$$

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

Sandia National Laboratories

Building a Status Test

- Converge if both: $\|F\| \leq 1.0E-6 \qquad \|\delta x\|_{WRMS} \leq 1.0$
- 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()`

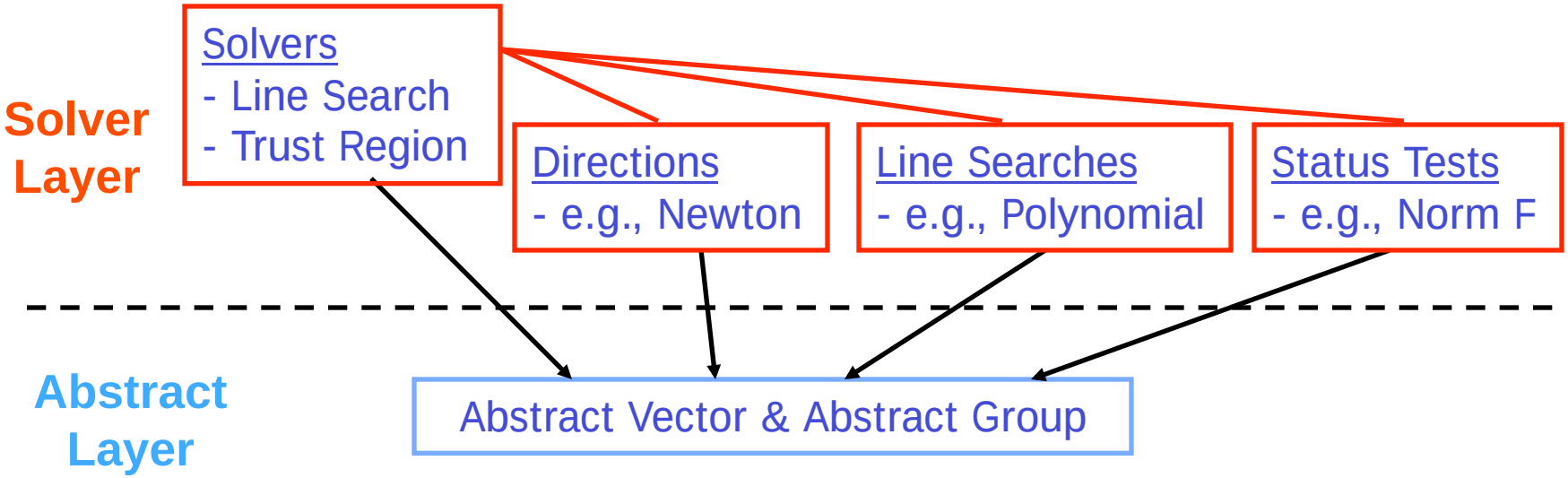
NOX solvers will work with any group/vector that implements these interfaces.

Four concrete implementations are supported:

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

Sandia National Laboratories

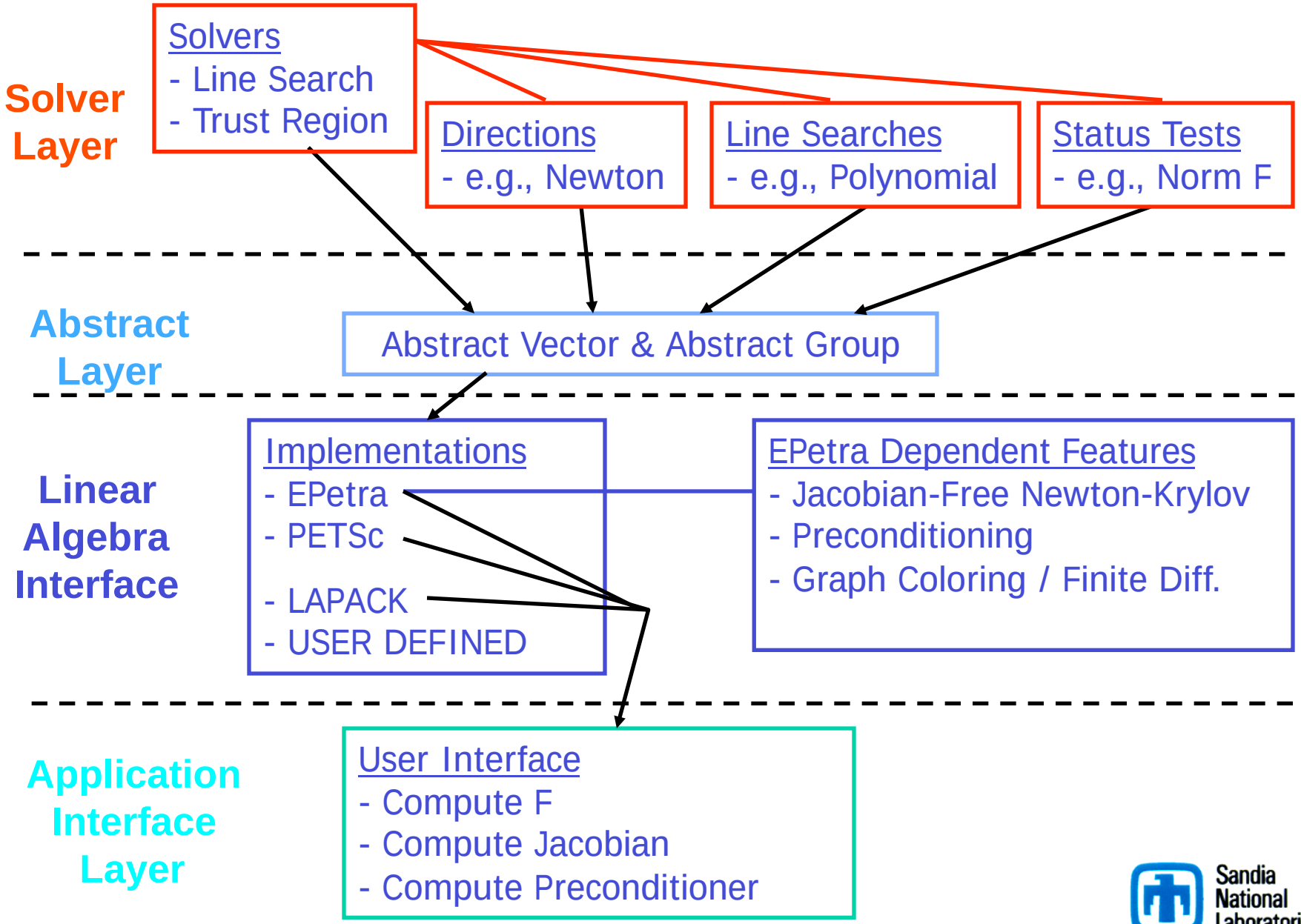
NOX Interface



- 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

Sandia National Laboratories