Trilinos Advanced Capabilities, Extensibility and Future Directions

Michael A. Heroux
Sandia National Laboratories

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.

Sandia
National
Laboratories

Trilinos Contributors (past 3 years)

Chris Baker
Developer of Anasazi, RBGen

Ross Bartlett
Lead Developer of MOOCHO, Stratimikos, RTOp, Thyra
Developer of Rythmos

Pavel Bochev
Lead Developer Intrepid

Paul Boggs
Developer of Thyra

Erik Boman
Lead Developer Isorropia
Developer Zoltan

Todd Coffey
Lead Developer of Rythmos

Karen Devine
Lead Developer Zoltan

Clark Dohrmann
Lead Developer of CLAPS

Carter Edwards
Lead Developer phdMesh

Michael Gee
Developer of ML, Moertel, NOX

Bob Heaphy
Lead developer of Trilinos SQA
Developer Zoltan

Mike Heroux
Trilinos Project Leader
Lead Developer of Epetra, AztecOO, Kokkos, IFPACK, Tpetra
Developer of Amesos, Belos, EpetraExt

Ulrich Hetmaniuk
Developer of Anasazi

Robert Hoekstra
Lead Developer of EpetraExt
Developer of Epetra

Russell Hooper
Developer of NOX

Vicki Howle
Lead Developer of Meros

Jonathan Hu
Developer of ML

Joe Kotulski
Lead Developer of Pliris

Rich Lehoucq
Developer of Anasazi and Belos

Kevin Long
Developer of Thyra

Roger Pawlowski
Lead Developer of NOX

Michael Phenow
Trilinos Webmaster
Developer WebTrilinos

Eric Phipps
Lead developer Sacado
Developer of LOCA, NOX

Dennis Ridzal
Lead Developer of Aristos, Intrepid

Marzio Sala
Lead Developer of Didasko, Galeri, IFPACK, WebTrilinos
Developer of ML, Amesos

Andrew Salinger
Lead Developer of LOCA, Capo

Paul Sexton
Developer of Epetra and Tpetra

Bob Shuttleworth
Developer of Meros.

Chris Siefert
Developer of ML

Bill Spotz
Lead Developer of PyTrilinos
Developer of Epetra, New_Package

Ken Stanley
Lead Developer of Amesos and New_Package

Heidi Thornquist
Lead Developer of Anasazi, Belos, RBGen and Teuchos

Ray Tuminaro
Lead Developer of ML and Meros

Jim Willenbring
Developer of Epetra and New_Package.
Trilinos library manager

Alan Williams
Lead Developer Isorropia, FEI
Developer of Epetra, EpetraExt, AztecOO, Tpetra

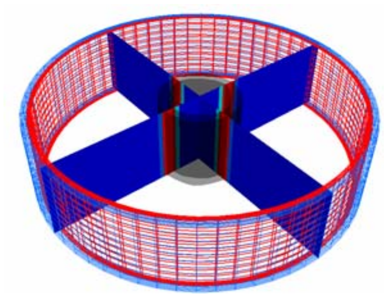
Take Home Messages

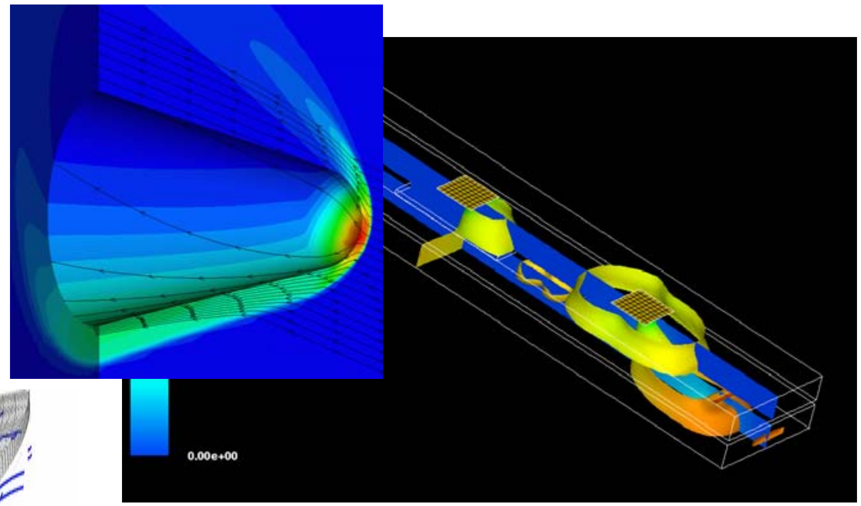
- Trilinos is both:
  - A development community
  - A collection of software
- OO techniques lead to:
  - Extensibility at many levels.
  - Scalable infrastructure.
  - Interoperability of independently developed capabilities.
  - Ability to adjust to architecture changes.
- Project is growing:
  - Including more of “vertical software stack”.
  - Adapting to broader user base.
- We are seeking collaborations with broader DOE community.

Sandia National Laboratories

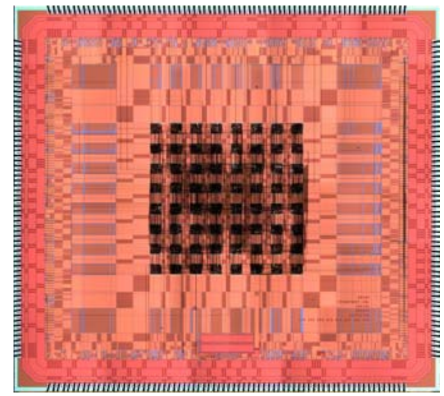
Background/Motivation

Target Problems: PDES and more…

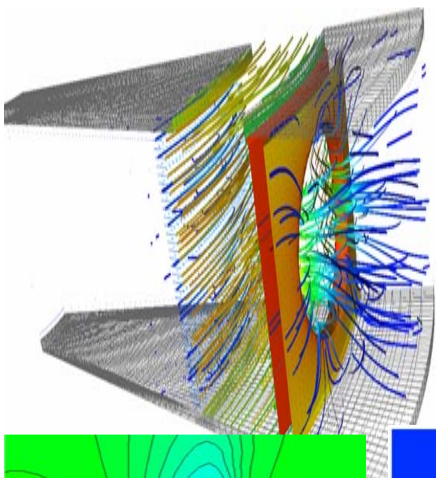




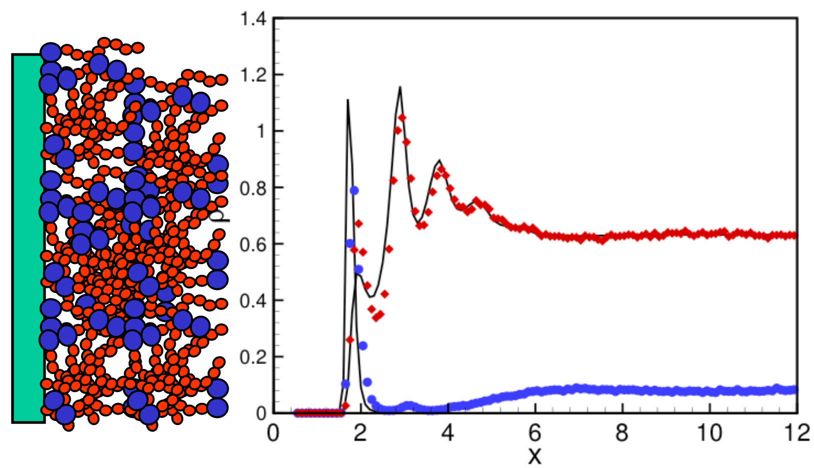
Circuits

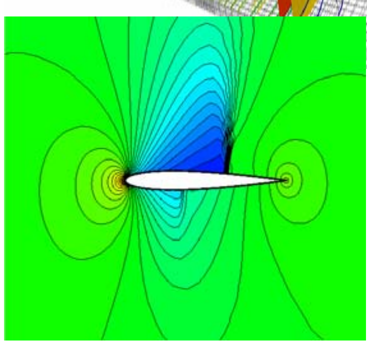


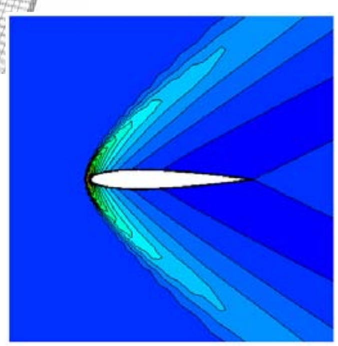
PDES



Inhomogeneous Fluids

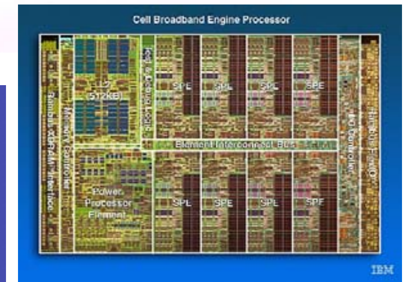






And More…

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



- Desktop: Development and more…
- Capability machines:
  - Redstorm (XT3), Clusters
  - Roadrunner (Cell-based).
  - Large-count multicore nodes.
- Parallel software environments:
  - MPI of course.
  - UPC, CAF, threads, vectors,…
  - Combinations of the above.
- User “skins”:
  - C++/C, Python
  - Fortran.
  - Web, CCA.

Motivation For Trilinos

- Sandia does LOTS of solver work.
- When I started at Sandia in May 1998:
  - Aztec was a mature package. Used in many codes.
  - FETI, PETSc, DSCPack, Spooles, ARPACK, DASPK, and many other codes were (and are) in use.
  - New projects were underway or planned in multi-level preconditioners, eigensolvers, non-linear solvers, etc…
- The challenges:
  - Little or no coordination was in place to:
    - Efficiently reuse existing solver technology.
    - Leverage new development across various projects.
    - Support solver software processes.
    - Provide consistent solver APIs for applications.
  - ASCI (now ASC) was forming software quality assurance/engineering (SQA/SQE) requirements:
    - Daunting requirements for any single solver effort to address alone.

Sandia National Laboratories

Evolving Trilinos Solution

- Trilinos[1] is an evolving framework to address these challenges:
  - Fundamental atomic unit is a *package*.
  - Includes core set of vector, graph and matrix classes (Epetra/Tpetra packages).
  - Provides a common abstract solver API (Thyra package).
  - Provides a ready-made package infrastructure (new_package package):
    - Source code management (cvs, bonsai).
    - Build tools (autotools).
    - Automated regression testing (queue directories within repository).
    - Communication tools (mailman mail lists).
  - Specifies requirements and suggested practices for package SQA.
- In general allows us to categorize efforts:
  - Efforts best done at the Trilinos level (useful to most or all packages).
  - Efforts best done at a package level (peculiar or important to a package).
  - **Allows package developers to focus only on things that are unique to their package.**

1. Trilinos loose translation: "A string of pearls"

Sandia National Laboratories

Evolving Trilinos Solution

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

physics

$L(u)=f$
Math. model

$L_h(u_h)=f_h$
Numerical model

$u_h=L_h^{-1} \bullet f_h$
Algorithms

computation

Numerical math
Convert to models that can be solved on digital computers

Algorithms
Find faster and more efficient ways to solve numerical models

discretizations
Time domain
Space domain

methods
Automatic diff.
Domain dec.
Mortar methods

Trilinos

solvers
Linear
Nonlinear
Eigenvalues
Optimization

core
Petra
Utilities
Interfaces
Load Balancing

- Discretization methods, AD, Mortar methods, …

Sandia National Laboratories

Trilinos Package Concepts

Package: The Atomic Unit

Trilinos Packages

- Trilinos is a collection of *Packages*.
- Each package is:
  - Focused on important, state-of-the-art algorithms in its problem regime.
  - Developed by a small team of domain experts.
  - Self-contained: No explicit dependencies on any other software packages (with some special exceptions).
  - Configurable/buildable/documented on its own.
- Sample packages: NOX, AztecOO, ML, IFPACK, Meros.
- Special package collections:
  - Petra (Epetra, Tpetra, Jpetra): Concrete Data Objects
  - Thyra: Abstract Conceptual Interfaces
  - Teuchos: Common Tools.
  - New_package: Jumpstart prototype.

Sandia National Laboratories

Trilinos Package Summary

	Objective	Package(s)
Discretizations	Meshing & Spatial Discretizations	phdMesh, Intrepid
	Time Integration	Rythmos
Methods	Automatic Differentiation	Sacado
	Mortar Methods	Moertel
Core	Linear algebra objects	Epetra, Jpetra, Tpetra
	Abstract interfaces	Thyra, Stratimikos, RTOp
	Load Balancing	Zoltan, Isorropia
	“Skins”	PyTrilinos, WebTrilinos, Star-P, *ForTrilinos*
	C++ utilities, (some) I/O	Teuchos, EpetraExt, Kokkos, Triutils
Solvers	Iterative (Krylov) linear solvers	AztecOO, Belos, Komplex
	Direct sparse linear solvers	Amesos
	Direct dense linear solvers	Epetra, Teuchos, Pliris
	Iterative eigenvalue solvers	Anasazi
	ILU-type preconditioners	AztecOO, IFPACK
	Multilevel preconditioners	ML, CLAPS
	Block preconditioners	Meros
	Nonlinear system solvers	NOX, LOCA
	Optimization (SAND)	MOOCHO, Aristos

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: < 2% total SLOC.

- Trilinos is not “indivisible”:
  - You don’t need all of Trilinos to get things done.
  - Any collection of packages can be combined and distributed.
  - Current public release contains only 30 of the 40+ Trilinos packages.

Insight from History

A Philosophy for Future Directions

- In the early 1800's U.S. had many new territories.
- Question: How to incorporate into U.S.?
  - Colonies? No.
  - Expand boundaries of existing states? No.
  - Create process for self-governing regions. Yes.
  - Theme: Local control drawing on national resources.
- Trilinos package architecture has some similarities:
  - Asynchronous maturation.
  - Packages decide degree of interoperations, use of Trilinos facilities.
- Strength of each: Scalable growth with local control.

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.

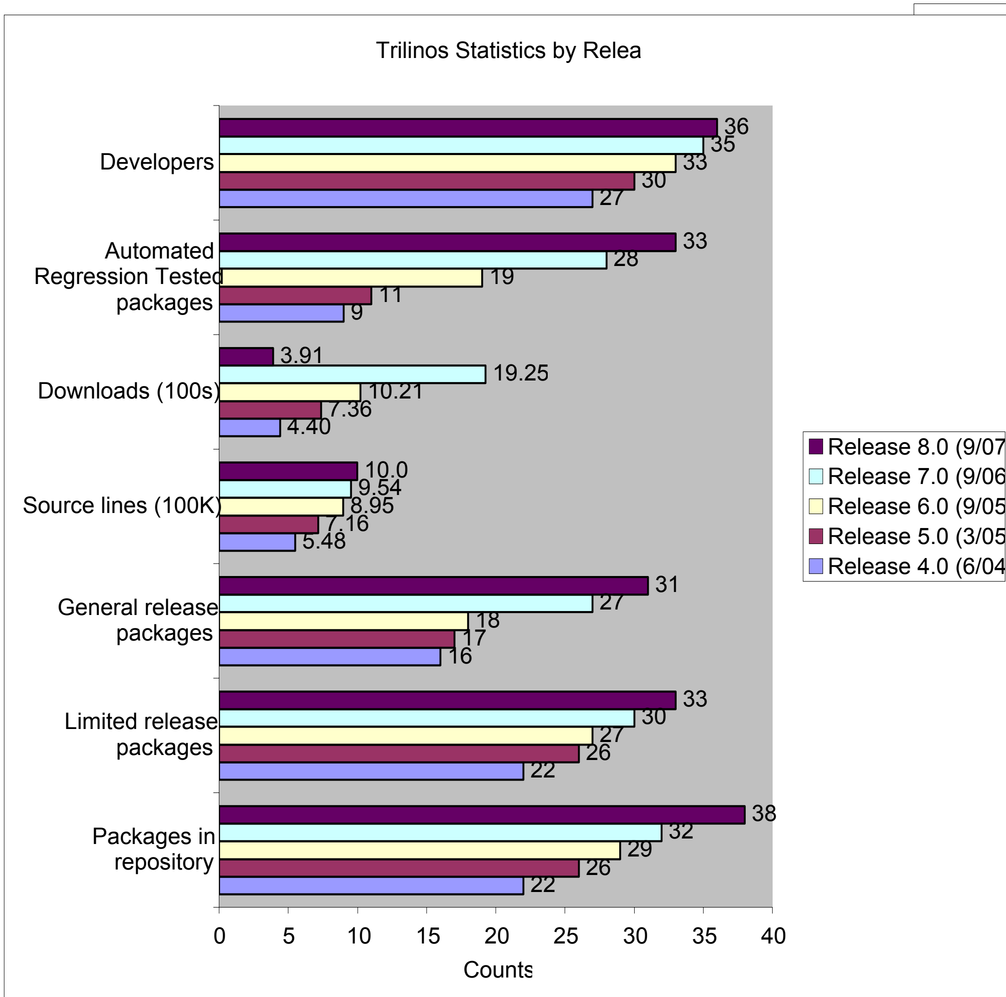
Algorithmic Goals

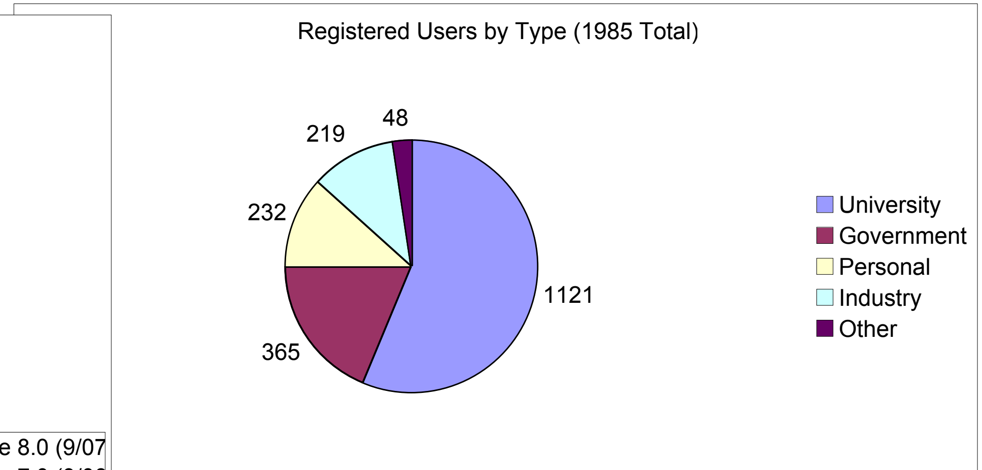
- **Grand Universal Interoperability:** All Trilinos packages will be interoperable, so that any combination of packages that makes sense algorithmically will be possible within Trilinos and with compatible external software.
- **Universal Accessibility:** All Trilinos capabilities will be available 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:
  - Integrated into every major application at Sandia (**Availability**).
  - The leading edge hardened, efficient, scalable solution for each of these applications (**Reliability**).
  - Easy to maintain and upgrade within the application environment (**Serviceability**).

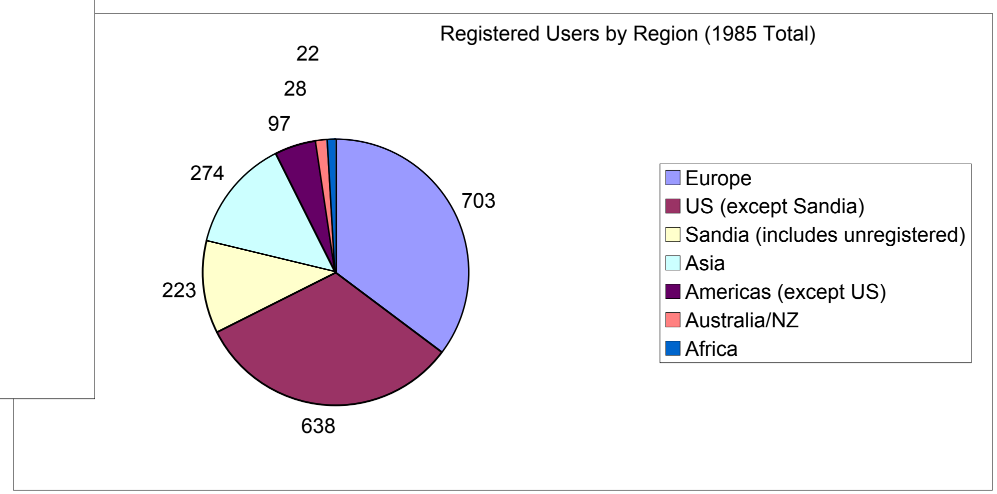
Software Goals

Sandia National Laboratories

Trilinos Statistics







Stats: Trilinos Download Page 10/16/2007.

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: Boeing, Goodyear, ExxonMobil.
- Linux distros: Debian, Mandriva.
- Star-P Interface.
- SciDAC TOPS-2 partner.
- Over 5000 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 Presentation Forums

- Next Trilinos User Group Meeting:
  - Nov 6-8, 2007.
  - At Sandia National Laboratories, Albuquerque, NM, USA.
- ACTS "Hands-on" Tutorial:
  - Aug 19-21, 2008.
  - At Lawrence Berkeley Lab, Berkeley, CA, USA.

Sandia National Laboratories

Website

- http:trilinos.sandia.gov Developer content on software.sandia.gov.
- Always looking to improve layout, content.
- Site was recently redesigned.

Whirlwind Tour of Packages

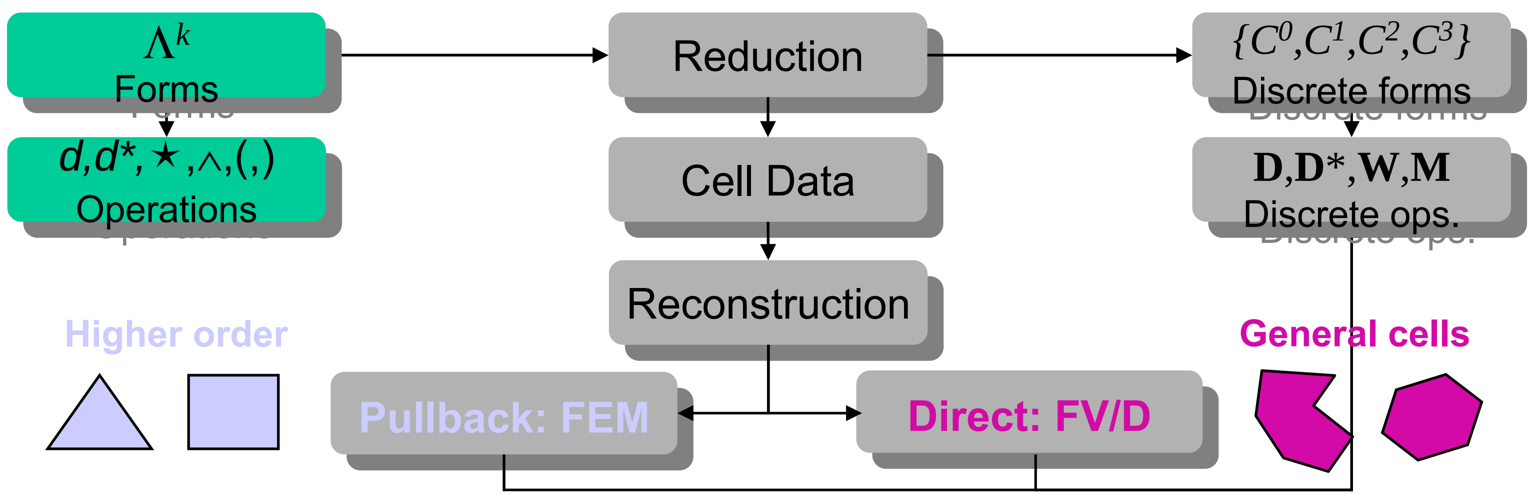
Discretizations **Methods** **Core** **Solvers**

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 reconstructions
- 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, Denis Ridzal, David Day

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

Sandia National Laboratories

Moertel: Mortar Methods

- Capabilities for nonconforming mesh tying and contact formulations in 2 and 3 dimensions using Mortar methods.

- Mortar methods are types of Lagrange Multiplier constraints that can be used in contact formulations and in non-conforming or conforming mesh tying as well as in domain decomposition techniques.

- Used in a large class of nonconforming situations such as the surface coupling of different physical models, discretization schemes or non-matching triangulations along interior interfaces of a domain.

Developer: Michael Gee

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

Trilinos

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)
- ParameterList:
  - Allows easy control of solver parameters.
  - XML format input/output.

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

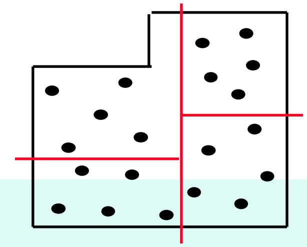
Kokkos

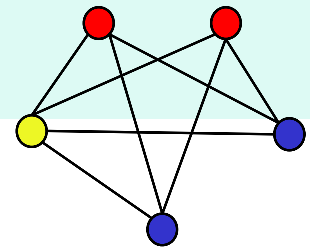
- Collection of several sparse/dense kernels that affect the performance of preconditioned Krylov methods
- Goal:
  - Isolate key non-BLAS kernels for the purposes of optimization.
- Kernels:
  - Dense vector/multivector updates and collective ops (not in BLAS/Teuchos).
  - Sparse MV, MM, SV, SM.
- Serial-only for now.
- Reference implementation provided (templated).
- Mechanism for improving performance:
  - Default is aggressive compilation of reference source.
  - BeBOP: Jim Demmel, Kathy Yelick, Rich Vuduc, UC Berkeley.
  - Vector version: Cray.

Developer: Mike Heroux

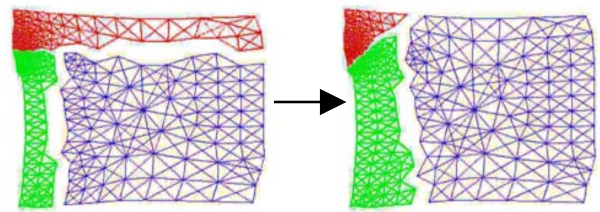
Sandia National Laboratories

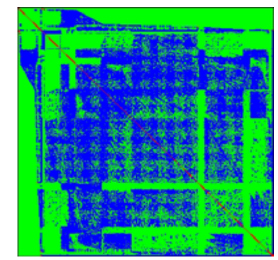
Zoltan



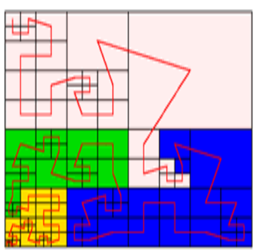


- Data Services for Dynamic Applications
  - Dynamic load balancing
  - Graph coloring
  - Data migration
  - Matrix ordering

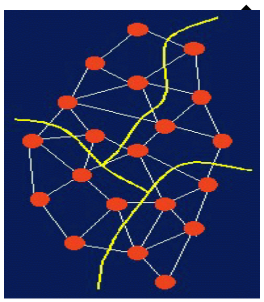




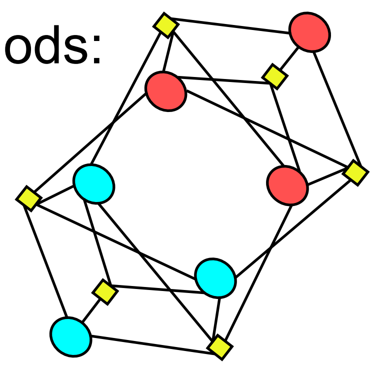
- Partitioners:



  - 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 Partitioning
    - Hypergraph Repartitioning PaToH (Catalyurek)
    - Zoltan Hypergraph Partitioning
    - ParMETIS (U. Minnesota)
    - Jostle (U. Greenwich)



Developers: Karen Devine, Eric Boman, Robert Heaphy

Sandia National Laboratories

Trilinos Common Language: Petra

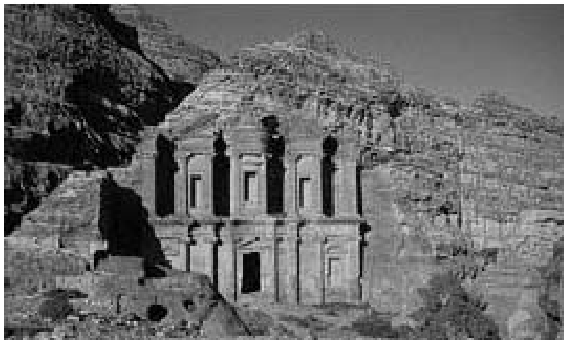
- Petra provides a “common language” for distributed linear algebra objects (operator, matrix, vector)

- 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: Mike Heroux, Rob Hoekstra, Alan Williams, Paul Sexton

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

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, ML, PETSc, LAPACK
- Transient/DAE solvers (implicit)
  - Abstraction of nonlinear solve.
  - … and so on.

Developers: Roscoe Bartlett, Kevin Long

Sandia National Laboratories

PyTrilinos

- PyTrilinos provides Python access to Trilinos packages.
- Uses SWIG to generate bindings.
- Epetra, AztecOO, IFPACK,
  ML, NOX, LOCA, Amesos
  and NewPackage are support.
- Possible to:
  - Define RowMatrix implementation in Python.
  - Use from Trilinos C++ code.
- Performance for large grain is equivalent to C++.
- Several times hit for very fine grain code.

Developer: Bill Spotz

WebTrilinos

- 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.

Developers: Ray Tuminaro, Jonathan Hu, Marzio Sala

Sandia National Laboratories

Trilinos

Whirlwind Tour of Packages

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

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

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

Belos and Anasazi

- Next generation linear solvers (Belos) and eigensolvers (Anasazi) libraries, written in templated C++.

- Provide a generic interface to a collection of algorithms for solving large-scale linear problems and eigenproblems.
- Algorithm implementation is accomplished through the use of abstract base classes (mini interface) and traits classes. Interfaces are derived from these base classes to:
  - operator-vector products
  - status tests
  - orthogonalization
  - any arbitrary linear algebra library.
- Includes block linear solvers and eigensolvers.

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

Sandia National Laboratories

NOX: Nonlinear Solvers

- Suite of nonlinear solution methods

- NOX uses abstract vector and “group” interfaces:
  - Allows flexible selection and tuning of various strategies:
    - Directions.
    - Line searches.
  - Epetra/AztecOO/ML, LAPACK, PETSc implementations of abstract vector/group interfaces.
- Designed to be easily integrated into existing applications.

Developers: Tammy Kolda, Roger Pawlowski

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

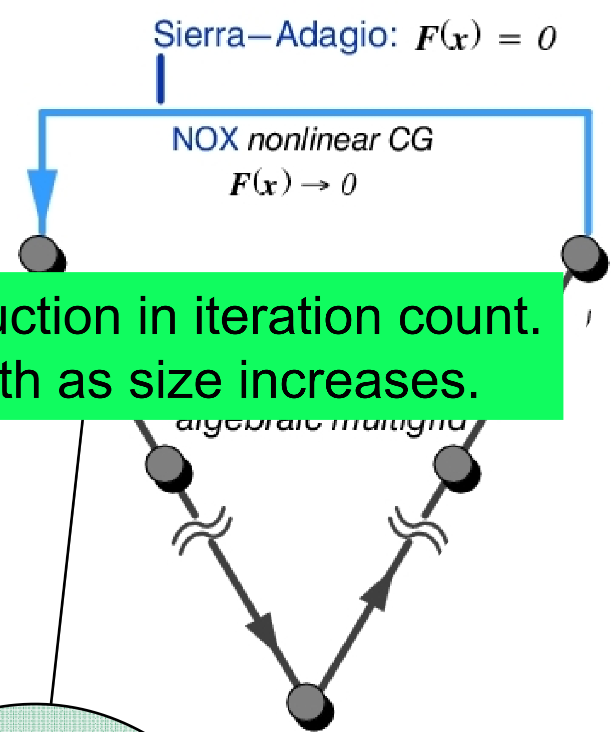
		Trilinos Packages
Optimization Problems: · Unconstrained: · Constrained:	Find $u \in \Re^n$ that minimizes $f(u)$ Find $y \in \Re^m$ and $u \in \Re^n$ that minimizes $f(y,u)$ s.t. $c(y,u)=0$	MOOCHO, Aristos
· Transient Problems: · DAEs/ODEs:	Solve $f(\dot{x}(t), x(t), t) = 0$ $t \in [0,T],\; x(0) = x_0,\; \dot{x}(0) = x_0'$ for $x(t) \in \Re^n,\; t \in [0,T]$	Rythmos
· Nonlinear Problems: · Nonlinear equations: · Stability analysis:	Given nonlinear op $c(x,u) \in \Re^{n+m} \rightarrow \Re^n$ Solve $c(x)=0$ for $x \in \Re^n$ For $c(x,u)=0$ find space $u \in U \ni \frac{\partial c}{\partial x}$ singular	NOX LOCA
· Implicit Linear Problems: · Linear equations: · Eigen problems:	Given linear ops (matrices) $A, B \in \Re^{n \times n}$ Solve $Ax = b$ for $x \in \Re^n$ Solve $Av = \lambda Bv$ for (all) $v \in \Re^n,\; \lambda \in \Re$	AztecOO, Belos, Ifpack,ML,etc. Anasazi
· Explicit Linear Problems: · Matrix/graph equations: · Vector problems:	Compute $y = Ax;\; A = A(G);\; A \in \Re^{m \times n}, G \in \square^{m \times n}$ Compute $y = \alpha x + \beta w;\; \alpha = \langle x, y \rangle;\; x, y \in \Re^n$	Epetra, Tpetra

Sandia National Laboratories

Algorithms Research: Truly Useful Multi-level Methods

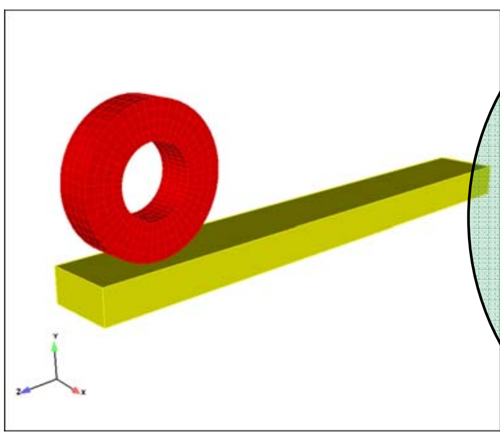
- Fly-through of next 4 slides.
- Theme:
  Multi-level preconditioning has come of age across broad spectrum of problems.

Nonlinear AMG/ADAGIO



- 1542/1414 **(Gee,Heinstein,Key,Pierson,Tuminaro)**
- Novel nonlinear AMG
  - ML, NOX, Epetra, Amesos, AztecOO
- Constraints projected out in F(x)
- Improved coloring performance by 10x
- Initial testing ⇒ excellent convergence

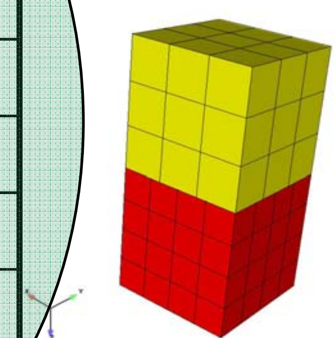
• Large reduction in iteration count.
• Slow growth as size increases.



Pressurize tire with 5 loadsteps

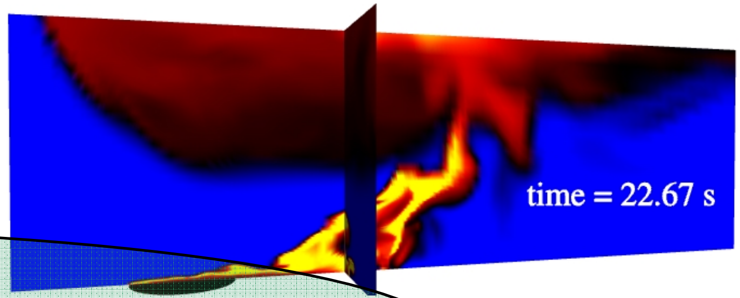
	its
Jacobi	22
MG	13

# elmts	its
91	13
559	14
1241	25
2331	23
3925	30
6119	35



ASC SIERRA Applications

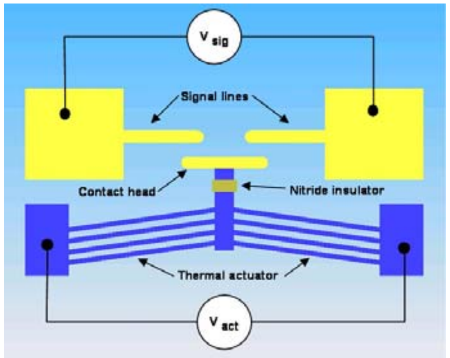
SIERRA/Fuego/Syrinx

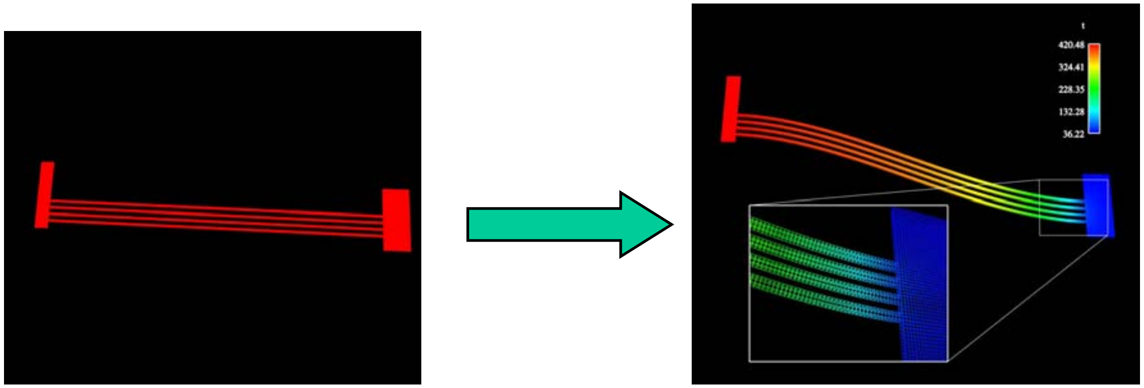


- Helium plume V&V Project
- 260K, 16 processor run
- ML/GMRES is ~25% faster than without ML
  - As problem size increase, ML expected to be more beneficial

SIERRA/Aria Multiphysics

- coupled potential/thermal/displacement DP MEMS problem
- ML reduced solve time (40%) from ~20 minutes to ~12 minutes
  - compared to actual ANSYS runs & ARIA re-creation ANSYS scheme





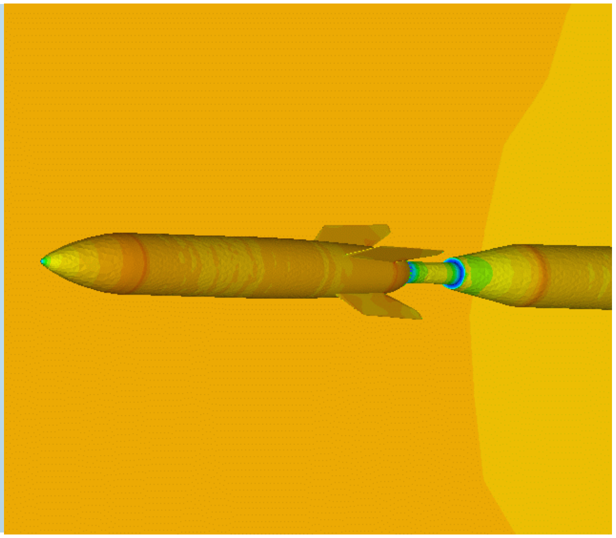
Premo

premo (Latin) – to squeeze (compress)

Compressible flow to determine aerodynamic characteristics for the Nuclear Weapons Complex

- Additional Issues that have been addressed
  - FEI/Nox/Trilinos interface development
  - Block Algorithm Improvements
  - Block Gauss-Seidel, Block Grid Transfers

- B61 problem (6.5 million dofs, 64 procs)
- Pseudo-transient + Newton
- Euler flow, Mach .8
- Linear solver: 173 solves, tolerance= 10^{-4}
- GMRES/ILU ➔ 7494 sec
- GMRES/MG ➔ 3704 sec

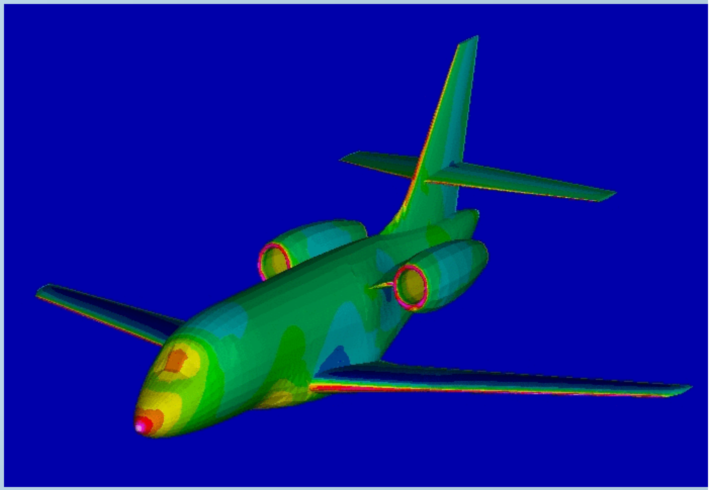


Premo

premo (Latin) – to squeeze (compress)

- Falcon problem (13^+ Million dofs, 150 procs)
- Pseudo-transient + Newton
- Euler flow, Mach .75
- Linear solver: 109 solves, tolerance= 10^{-4}

- GMRES/ILU ➔ 7620 sec
- GMRES/MG ➔ 3787 sec
- 10x improvement on final linear solve.
- > 5x gains on some problems over entire sequence



- New Grid Transfers (220^+K Falcon, 1 proc)
- Pseudo-transient + Newton, Euler flow @ Mach .75
- Last linear solve, tolerance= 10^{-9}

- GMRES/MG/old transfers ➔ 47 its, 49 sec
- GMRES/MG/new transfers ➔ 24 its, 26 sec

Sandia National Laboratories

Multi-level Methods Summary

- Solving hard, real problems fast, scalably.
- Still need more…

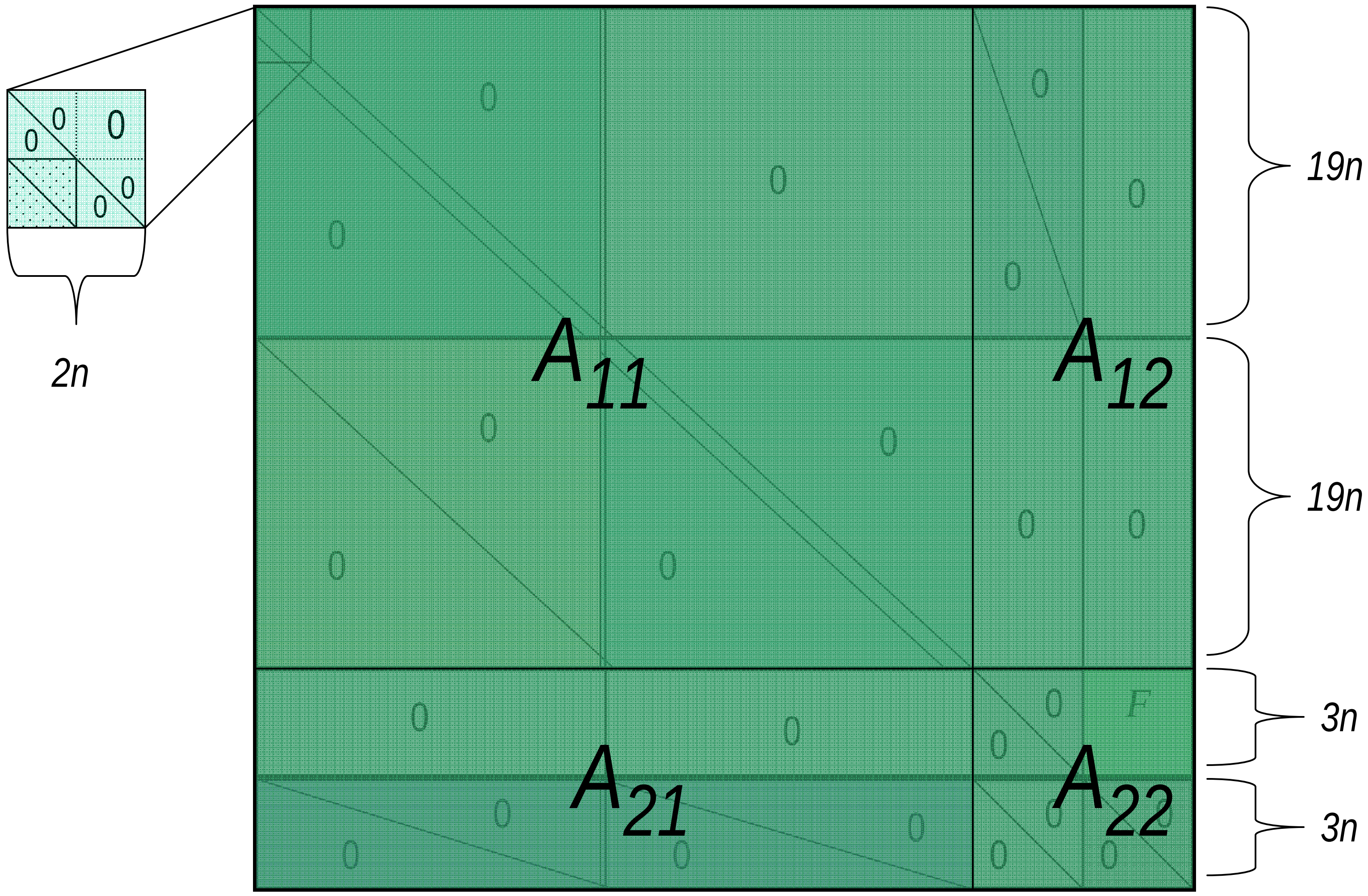
Algorithms Research: Specialized Solvers

- Next wave of capabilities: Specialized solvers.
- Examples in Trilinos:
  - Optimal domain decomposition preconditioners for structures: CLAPS
  - Mortar methods for interface coupling: Moertel.
  - Segregated Preconditioners for Navier-Stokes: Meros.
- Examples in Applications:
  - EMU (with Boeing).
  - Tramonto.

Sandia National Laboratories

- A_{11} solve easily applied in parallel.
- Apply GMRES to $S = A_{22} - A_{21} * inv(A_{11}) * A_{12}$
- Need a preconditioner for S.

Lipid Bi-Layer Problem: One example (of many variations)



Preconditioner for *S*

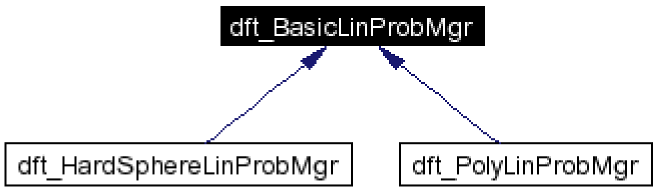
$$A_{22} = \begin{bmatrix} D_{11} & F \\ D_{21} & D_{21} \end{bmatrix}$$

$$A_{22} \approx \tilde{A}_{22} = \begin{bmatrix} D_{11} & F \\ 0 & D_{21} \end{bmatrix}$$

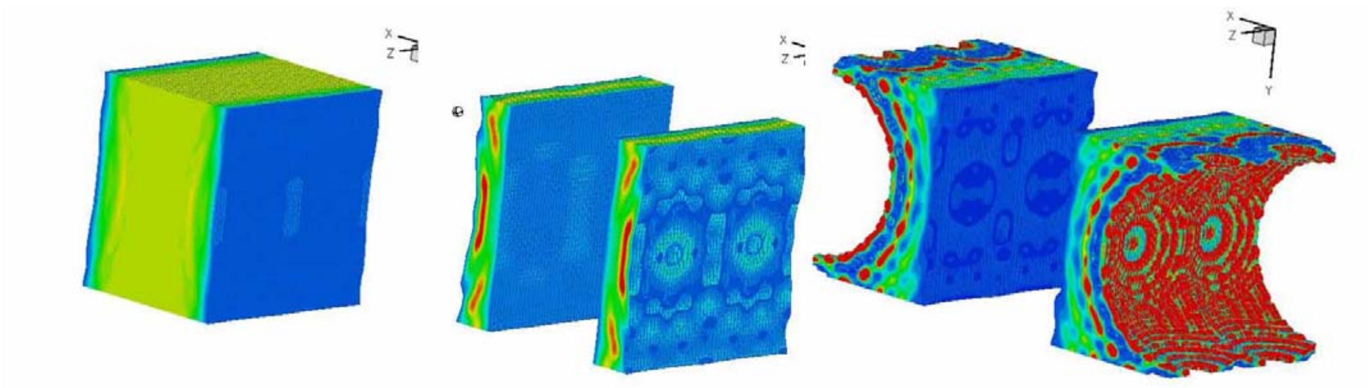
- D_{11}, $D_{22} = O(1)$, $D_{21} = O(1e\text{-}10)$
- Ignore D_{21} for preconditioning.
- *P(S)* requires
  - 2 diagonal scalings,
  - matvec with F.
- All distributed operations.

Tramonto Solver Summary

- 3-level linear operator structure.
- Matlab to fully parallel: 1 month.
- Complex orchestration:
  - Preconditioner: 100+ distributed Epetra matrices used in sequence.
  - ML, IFPACK, Amesos used on subproblems.
- Utilizes 8 Trilinos packages in total.
- 566 Lines of Code (Polymer Solver).
- Polymorphic Design.



3D Polymer Results



#Procs	Niter Old/New	<# Lin iters> Old	<# Lin iters> New	Solve Time Old	Solve Time New	$T_{100Proc}/T$ Old	$T_{100Proc}/T$ New	T_{OLD}/T Ratio
32	-/12	-	110	-	9069	-	0.26	-
100	10*/12	147*	110	10,090*	2381	1	1	4.2*
200	10/12	167	110	2415	1269	4.2*	1.9	1.9

Approximately Linear Performance Improvement For Fixed Size Problem

Same Iteration Counts as Processor Count Increases

Properties of New Solver

- Uniform distributed mesh → uniform distributed work.
- Preconditioner sub-steps naturally parallel:
  - → Results invariant to processor count up to round-off.
- Preconditioner requires almost no extra memory:
  - → 4-10X reduction over previous approach.
- GMRES subspace and storage reduced 6X-10X or more.
- Speedup 20-2X.
- Solver has:
  - No tuning parameters.
  - Near linear scaling.

Laurie's Favorite Properties!

Michael A. Heroux, Laura J. D. Frink, and Andrew G. Salinger.
Schur complement based approaches to solving density functional theories for inhomogeneous fluids on parallel computers. SIAM J. Sci. Comput. 2007.

Sandia National Laboratories

Extending Capabilities: Preconditioners, Operators, Matrices

Illustrated using AztecOO as example

Epetra User Class Categories

◆ Sparse Matrices:	***RowMatrix***, (CrsMatrix, VbrMatrix, FECrsMatrix, FEVbrMatrix)
◆ Linear Operator:	***Operator*****:** (AztecOO, ML, Ifpack)
◆ Dense Matrices:	DenseMatrix, DenseVector, BLAS, LAPACK, SerialDenseSolver
◆ Vectors:	Vector, MultiVector
◆ Graphs:	CrsGraph
◆ Data Layout:	Map, BlockMap, LocalMap
◆ Redistribution:	Import, Export, LbGraph, LbMatrix
◆ Aggregates:	LinearProblem
◆ Parallel Machine:	***Comm***, (SerialComm, MpiComm, MpiSmpComm)
◆ Utilities:	Time, Flops

Sandia National Laboratories

LinearProblem Class

- A linear problem is defined by:
  - Matrix *A* :
    - An Epetra_RowMatrix or Epetra_Operator object. (often a CrsMatrix or VbrMatrix object.)
  - Vectors *x*, *b* : Vector objects.
- To call AztecOO, first define a LinearProblem:
  - Constructed from *A*, *x* and *b.*
  - Once defined, can:
    - Scale the problem (explicit preconditioning).
    - Precondition it (implicitly).
    - Change *x* and *b.*

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.

A Simple Epetra/AztecOO Program

```
// Header files omitted…
int main(int argc, char *argv[]) {
  MPI_Init(&argc,&argv); // Initialize MPI, MpiComm
  Epetra_MpiComm Comm( MPI_COMM_WORLD );
```

```
// ***** 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;

  MPI_Finalize() ;
  return 0;
}
```

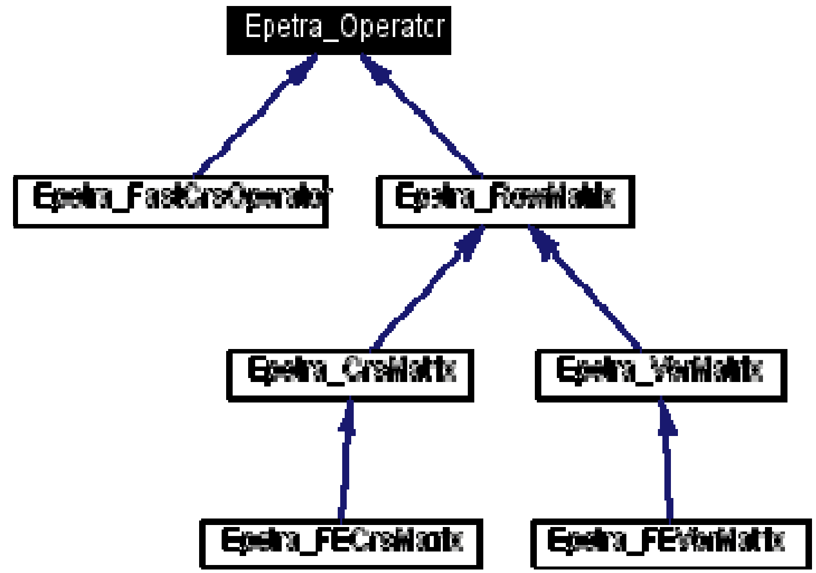
Sandia National Laboratories

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.

AztecOO understands Epetra_Operator

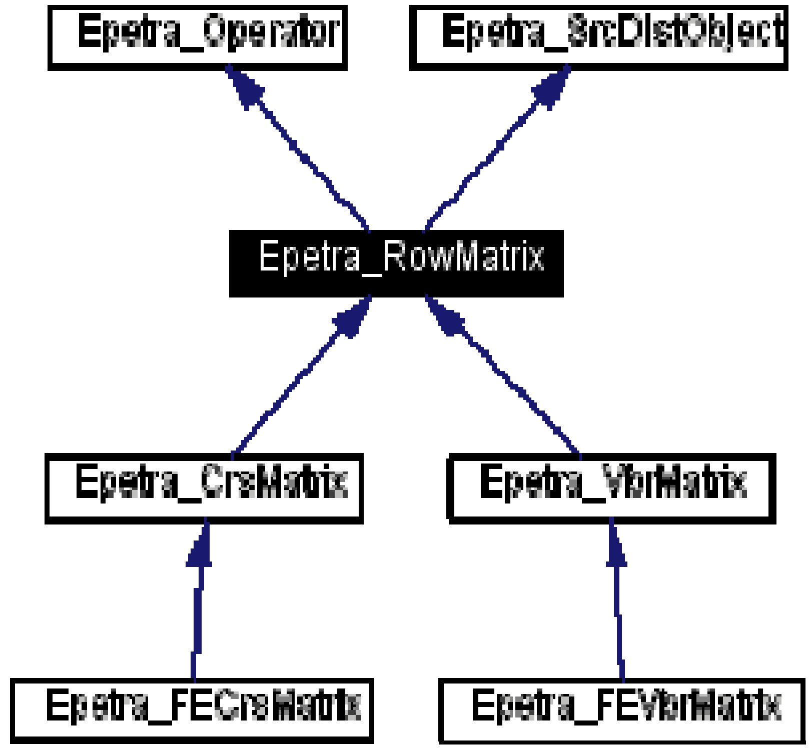
- AztecOO is designed to accept externally defined:
  - Operators (both *A* and *M*).
  - RowMatrix (Facilitates use of AztecOO preconditioners with external *A*).
  - StatusTests (externally-defined stopping criteria).



[Epetra_Operator Methods Documentation]

AztecOO Understands Epetra_RowMatrix

Epetra_RowMatrix Methods



AztecOO UserOp/UserMat Recursive Call Example

Trilinos/packages/aztecoo/example/AztecOO_RecursiveCall

```
Poisson2dOperator A(nx, ny, comm); // Generate nx by ny Poisson operator
Epetra_CrsMatrix * precMatrix = A.GeneratePrecMatrix(); // Build tridiagonal approximate Poisson

Epetra_Vector xx(A.OperatorDomainMap()); // Generate vectors (xx will be used to generate RHS b)
Epetra_Vector x(A.OperatorDomainMap());
Epetra_Vector b(A.OperatorRangeMap());

xx.Random(); // Generate exact x and then rhs b
A.Apply(xx, b);

// Build AztecOO solver that will be used as a preconditioner
Epetra_LinearProblem precProblem;
precProblem.SetOperator(precMatrix);
AztecOO precSolver(precProblem);
precSolver.SetAztecOption(AZ_precond, AZ_ls);
precSolver.SetAztecOption(AZ_output, AZ_none);
precSolver.SetAztecOption(AZ_solver, AZ_cg);
AztecOO_Operator precOperator(&precSolver, 20);

 Epetra_LinearProblem problem(&A, &x, &b); // Construct linear problem
 AztecOO solver(problem); // Construct solver

solver.SetPrecOperator(&precOperator); // Register  Preconditioner operator

solver.SetAztecOption(AZ_solver, AZ_cg);
solver.Iterate(Niters, 1.0E-12);
```

Ifpack/AztecOO Example

Trilinos/packages/aztecoo/example/IfpackAztecOO

```
1.     // Assume A, x, b are define, LevelFill and Overlap are specified

2.      Ifpack_IlukGraph IlukGraph(A.Graph(), LevelFill, Overlap);
3.      IlukGraph.ConstructFilledGraph();
4.      Ifpack_CrsRiluk ILUK (IlukGraph);
5.      ILUK.InitValues(A);
6.      assert(ILUK->Factor()==0);  // Note: All Epetra/Ifpack/AztecOO method return int err codes
7.      double Condest;
8.      ILUK.Condest(false, Condest); // Get condition estimate
9.      if (Condest > tooBig) {
10.        ILUK.SetAbsoluteThreshold(Athresh);
11.       ILUK.SetRelativeThreshold(Rthresh);
12.        Go back to line 4 and try again
13.     }
14.     Epetra_LinearProblem problem(&A, &x, &b); // Construct linear problem
15.     AztecOO solver(problem); // Construct solver

16.     solver.SetPrecOperator(&ILUK); // Register  Preconditioner operator

17.    solver.SetAztecOption(AZ_solver, AZ_cg);
18.    solver.Iterate(Niters, 1.0E-12);


19.   //  Once this linear solutions complete and the next nonlinear step is advanced,
20.   //  we will return to the solver, but only need to execute steps 5 on down…
```

Multiple Stopping Criteria

- Possible scenario for stopping an iterative solver:
  - Test 1: Make sure residual is decreased by 6 orders of magnitude.

  And

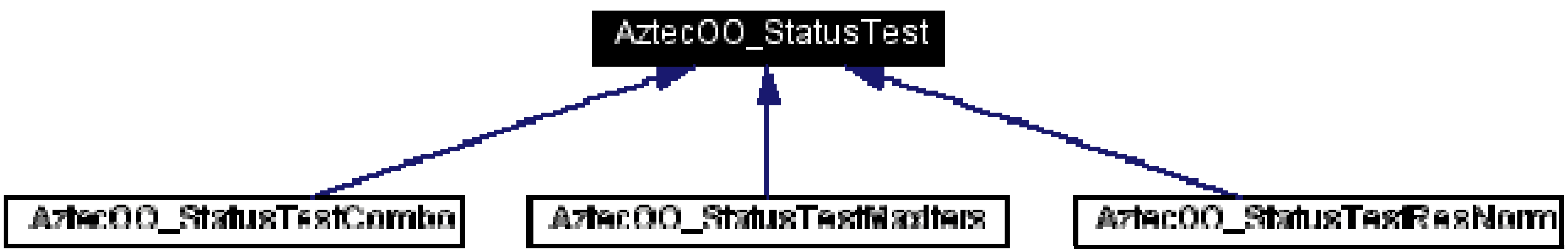
  - Test 2: Make sure that the inf-norm of <u>true</u> residual is no more 1.0E-8.

  But

  - Test 3: do no more than 200 iterations.
- Note: Test 1 is *cheap*. Do it before Test 2.

AztecOO StatusTest classes

- AztecOO_StatusTest:
  - Abstract base class for defining stopping criteria.
  - Combo class: OR, AND, SEQ



AztecOO_StatusTest Methods

Sandia National Laboratories

AztecOO/StatusTest Example

Trilinos/packages/aztecoo/example/AztecOO

```
 // Assume A, x, b are define

Epetra_LinearProblem problem(&A, &x, &b); // Construct linear problem
 AztecOO solver(problem); // Construct solver


AztecOO_StatusTestResNorm restest1(A, x, bb, 1.0E-6);
restest1.DefineResForm(AztecOO_StatusTestResNorm::Implicit, AztecOO_StatusTestResNorm::TwoNorm);
restest1.DefineScaleForm(AztecOO_StatusTestResNorm::NormOfInitRes, AztecOO_StatusTestResNorm::TwoNorm);

AztecOO_StatusTestResNorm restest2(A, x, bb, 1.0E-8);
restest2.DefineResForm(AztecOO_StatusTestResNorm::Explicit, AztecOO_StatusTestResNorm::InfNorm);
restest2.DefineScaleForm(AztecOO_StatusTestResNorm::NormOfRHS, AztecOO_StatusTestResNorm::InfNorm);

AztecOO_StatusTestCombo comboTest1(AztecOO_StatusTestCombo::SEQ, restest1, restest2);

AztecOO_StatusTestMaxIters maxItersTest(200);
AztecOO_StatusTestCombo comboTest2(AztecOO_StatusTestCombo::OR, maxItersTest1, comboTest1);
solver.SetStatusTest(&comboTest2);

  solver.SetAztecOption(AZ_solver, AZ_cg);
  solver.Iterate(Niters, 1.0E-12);
```

Summary: Extending Capabilities

- Trilinos packages are designed to interoperate.
- All packages (ML, IFPACK, AztecOO, …) that can provide linear operators:
  - Implement the Epetra_Operator interface.
  - Are available to any package that can use an linear operator.
- All packages (ML, AztecOO, NOX, Belos, Anasazi, …) that can use linear operators:
  - Accept linear operator via Epetra_Operator interface.
  - Support easy user extensions.
- All packages (ML, IFPACK, AztecOO, …) that need matrix coefficient data:
  - Can access that data from Epetra_RowMatrix interface.
  - Can use any concrete Epetra matrix class, or any user-provided adapter.

Sandia National Laboratories

Summary: Extending Capabilities

AztecOO is one example:

- Flexibility comes from abstract base classes:
  - Epetra_Operator:
    - All Epetra matrix classes implement.
    - Best way to define *A* and *M* when coefficient info **not** needed.
  - Epetra_RowMatrix:
    - All Epetra matrix classes implement.
    - Best way to define *A* and *M* when coefficient info **is** needed.
  - AztecOO_StatusTest:
    - A suite of parametrized status tests.
    - An abstract interface for users to define their own.
    - Ability to combine tests for sophisticated control of stopping.

Sandia National Laboratories

A Few More Useful Things

Fortran Interface

- Presently Trilinos has no full-featured Fortran interface.
- Plans in place to develop OO Fortran interface.
- Developed as part of SciDAC TOPS-2 effort.
- Just ramping up now.

Stratimikos

- New package in Trilinos 7.0.
- Single point of access to Trilinos preconditioners/solvers:
  - Common interface all preconditioners.
  - Common interface to all solvers.
  - Selection of preconditioner/solver via parameter list.
- Simplest way to access the suite of Trilinos capabilities.
- Simple driver code available on website.
- Will be the focus of Fortran access to Trilinos.

Dynamic External Package Support

- New directory Trilinos/packages/external.
- Supports seamless integration of externally developed packages via package registration.
- Your package: “WorldsBestPreconditioner”
  - Understands configure/make.
  - Can have its own options: --enable-superfast-mode
- Copy source into Trilinos/packages/external.
- In Trilinos/packages/external, type:
  ./CustomizeExternal.csh WorldsBestPreconditioner
- Build Trilinos in the usual way using configure/make.
  - Include arguments such as --enable-superfast-mode: They will be passed down to your package.

Software Quality

SQA/SQE

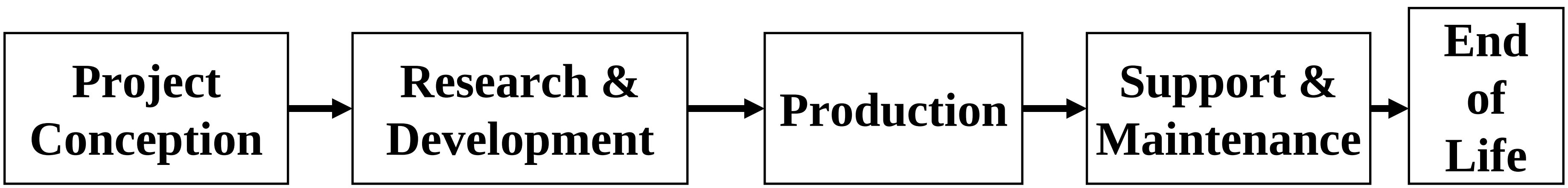
- Software Quality Assurance/Engineering is important.
- Not sufficient to say, “We do a good job.”
- Trilinos facilitates SQA/SQE development/processes for packages:
  - 10 of 30 ASC SQE practices are directly handled by Trilinos (no requirements on packages).
  - Trilinos provides infrastructure support for the remaining 20.
  - Trilinos Dev Guide Part II: Specific to ASC requirements.
  - Trilinos software engineering policies provide a ready-made infrastructure for new packages.
  - Trilinos philosophy:
    Few *requirements*. Instead mostly *suggested practices.* Provides package with option to provide alternate process.

Trilinos Service	SQE Practices Impact
Yearly Trilinos User Group Meeting (TUG) and Developer Forum: Once a year gathering for tutorials, package feature updates, user/developer requirements discussion and developer training.	— All Requirements steps: gathering, derivation, documentation, feasibility,etc. — User and Developer training.
Monthly Trilinos leaders meetings: Trilinos leaders, including package development leaders, key managers, funding sources and other stakeholders participate in monthly phone meetings to discuss any timely issues related to the Trilinos Project.	—Developer Training. —Design reviews. —Policy decisions across all development phases.
Trilinos and package mail lists: Trilinos lists for leaders, announcements, developers, users, checkins and similar lists at the package level support a variety of communication. All lists are archived, providing critical artifacts for assessments and audits.	—Developer/user/client communication. —Requirements/design/testing artifacts. —Announcement/documenting of releases.
Trilinos and Trilinos3PL source repositories: All source code, development and user documentation is retained and tracked. In addition, reference versions of all external software, including BLAS, LAPACK, Umfpack, etc. are retained in Trilinos3PL.	—Source management. —Versioning. —Third-party software management.
Bugzilla Products: Each package has its own Bugzilla Product with standard components.	—Requirements/faults capturing and tracking.
Trilinos configure script and M4 macros: The Trilinos configure script and related macros support portable installation of Trilinos and its packages	—Portability. —Software release.
Trilinos test harness: Trilinos provides a base testing plan and automated testing across multiple platforms, plus creation of testing artifacts. Test harness results are used to derive a variety of metrics for SQE.	—Pre-checkin and regression testing. —Software metrics.

Software Lifecycles

(Typical) Project Lifecycle

Consider this lifecycle



Scientific Research and Life Cycle Models

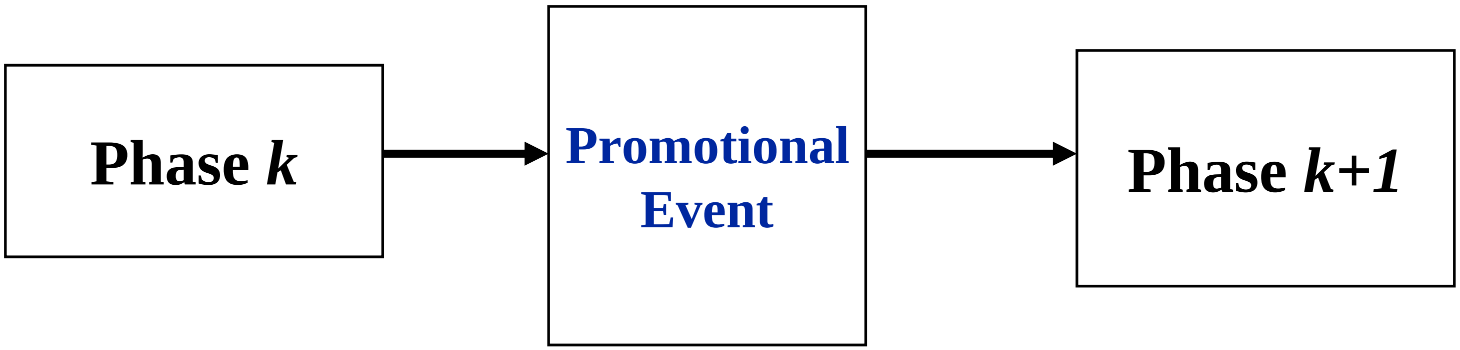
- Life Cycle Models are generally developed from the point of view of business software.
- Little consideration is given to algorithmic development.
- Traditional business execution environment is traditional mainframe or desktop, not parallel computers.
- Traditional development "techniques" are assumed.

Research Software needs a different model

- Research should be "informal":
  - Allow external collaborators, students, post-docs, etc.
  - Allow changes of direction without seeking permission
  - Should use modern software development paradigms
    - i.e. Lean/Agile methods
  - Must be verified more than validated

- Production code must:
  - Have formality appropriate to risks,
  - Be Complete (documentation, testing, …),
  - Be "user proofed",
  - Be platform independent (as necessary),
  - Be validated not just verified.

“Promotional” Model



- Lower formality
- Fewer Artifacts
- Lean/Agile

- Higher formality
- Sufficient Artifacts
- Bullet proof
- Maintainable

Trilinos Software Lifecycle Model

The Trilinos Software Lifecycle Model, James M. Willenbring and Michael A. Heroux and Robert T. Heaphy, *Proceeding of the 29th International Conference on Software Engineering*, May 2007

Trilinos Lifecycle Phases

- Three phases:
  - Research.
  - Production Growth.
  - Production Maintenance.
- Each phase contains its own lifecycle model.
- Promotional events:
  - Required for transition from one phase to next.
  - Signify change in behaviors and attitude.
- Phase assigned individually to each package.

Lifecycle Phase 1: Research

- Conducting research is the primary goal.
- Producing software is potentially incidental to research.
- Any software that is produced is typically a “proof of concept” or prototype.
- Software that is in this phase may only be released to selected internal customers to support their research or development and should not be treated as production quality code.

Phase 1 Required Practices

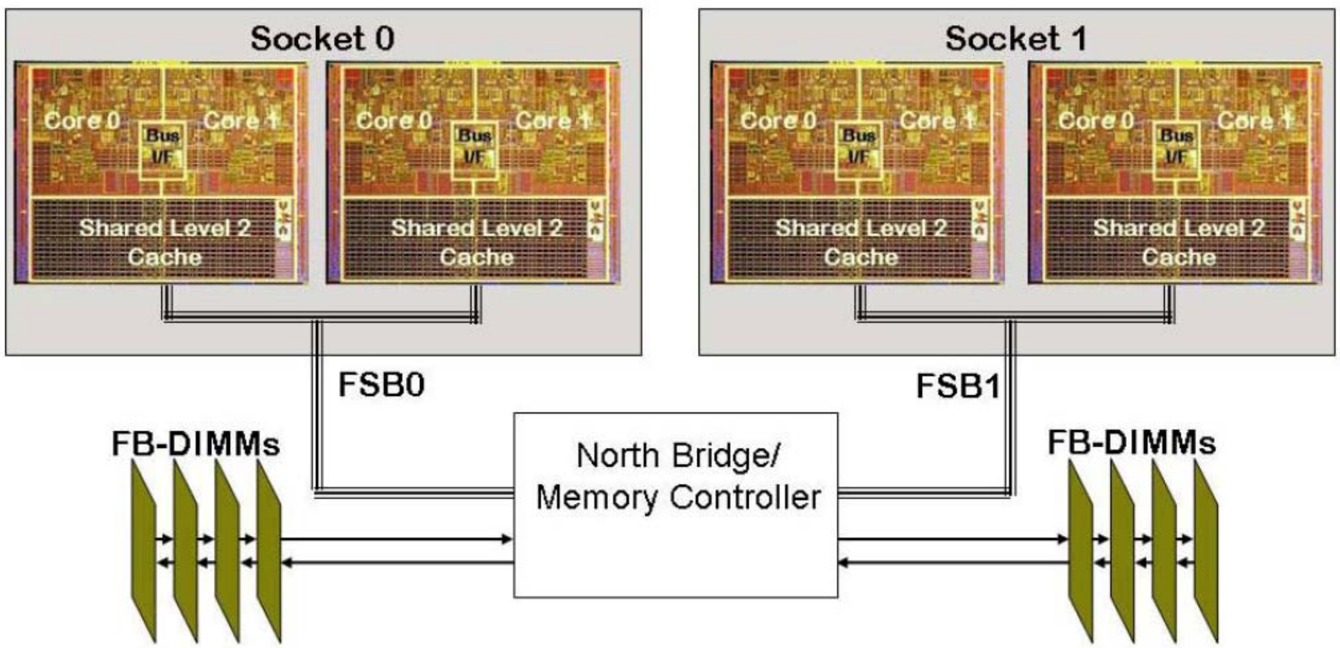
- The research proposal is the project plan.
- Software is placed under configuration control as needed to prevent loss due to disaster.
- Peer reviewed published papers are primary verification and validation.
- The focus of testing is a proof of correctness, not software.
- Periodic status reports should be produced, presentation is sufficient.
- A lab notebook, project notebook, or equivalent is the primary artifact.

Sandia National Laboratories

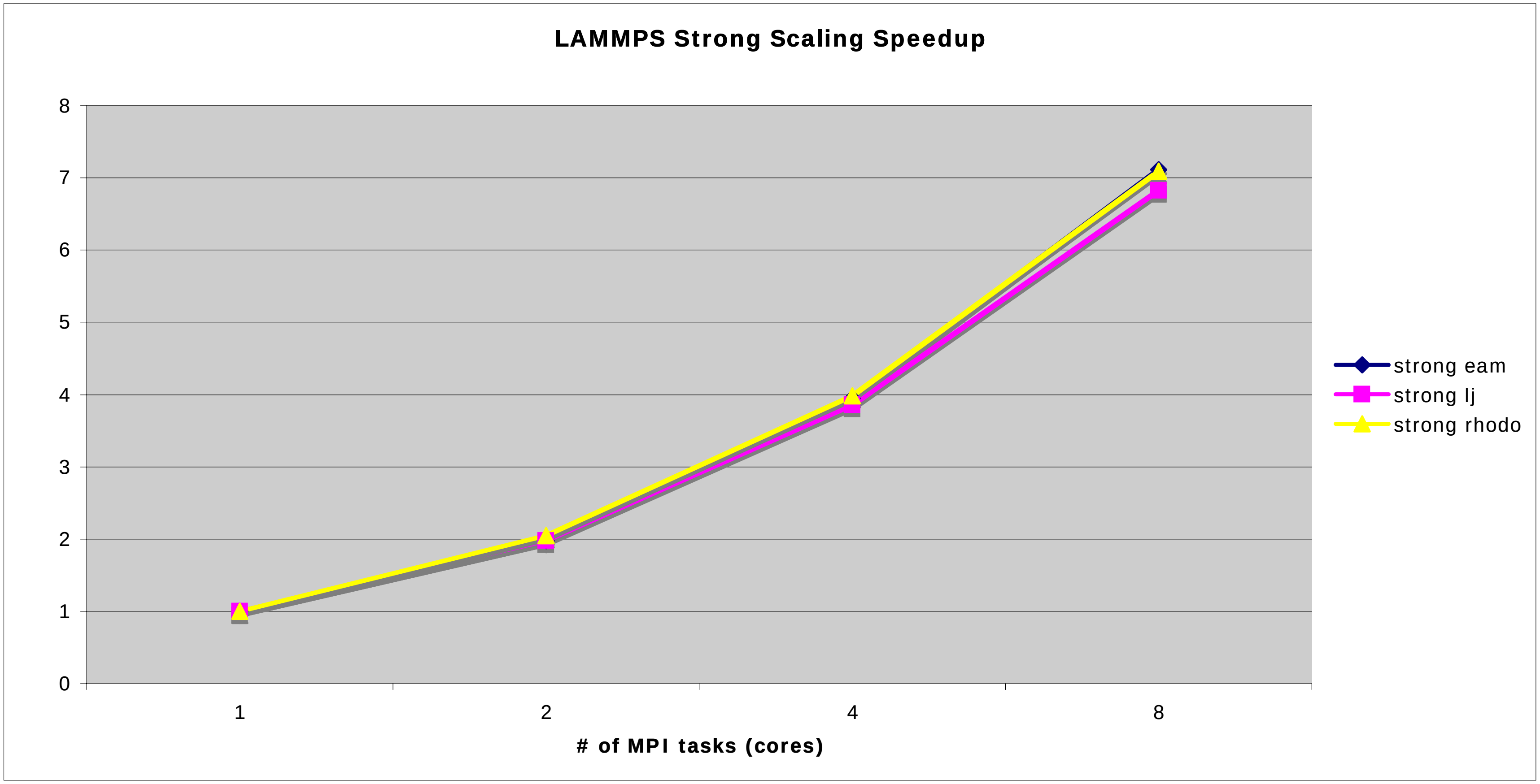
Multicore Efforts

Test Platform: Clovertown

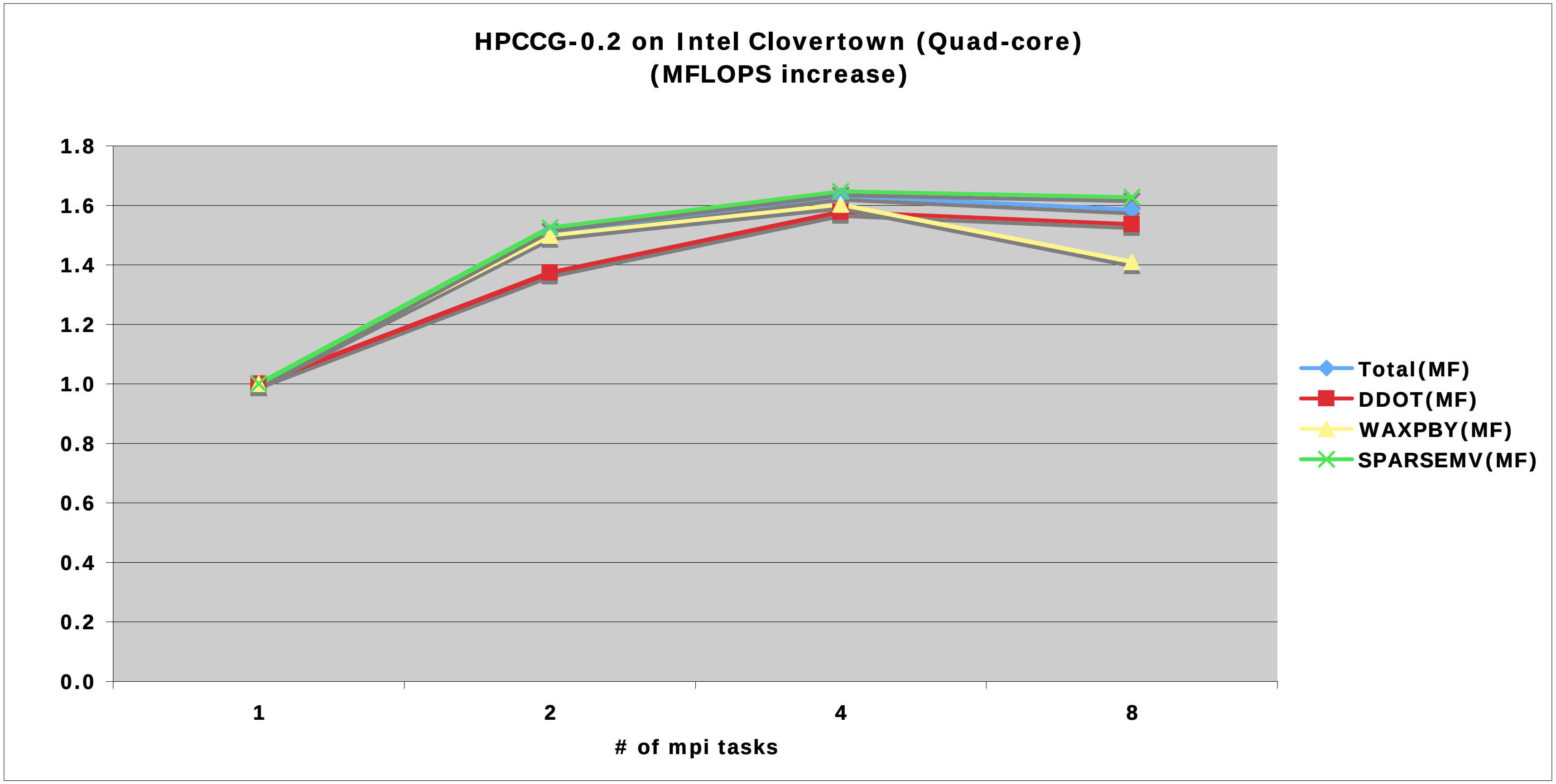
- Intel: Clovertown, Quad-core (actually two dual-cores)
- Performance results are based on 1.86 GHz version



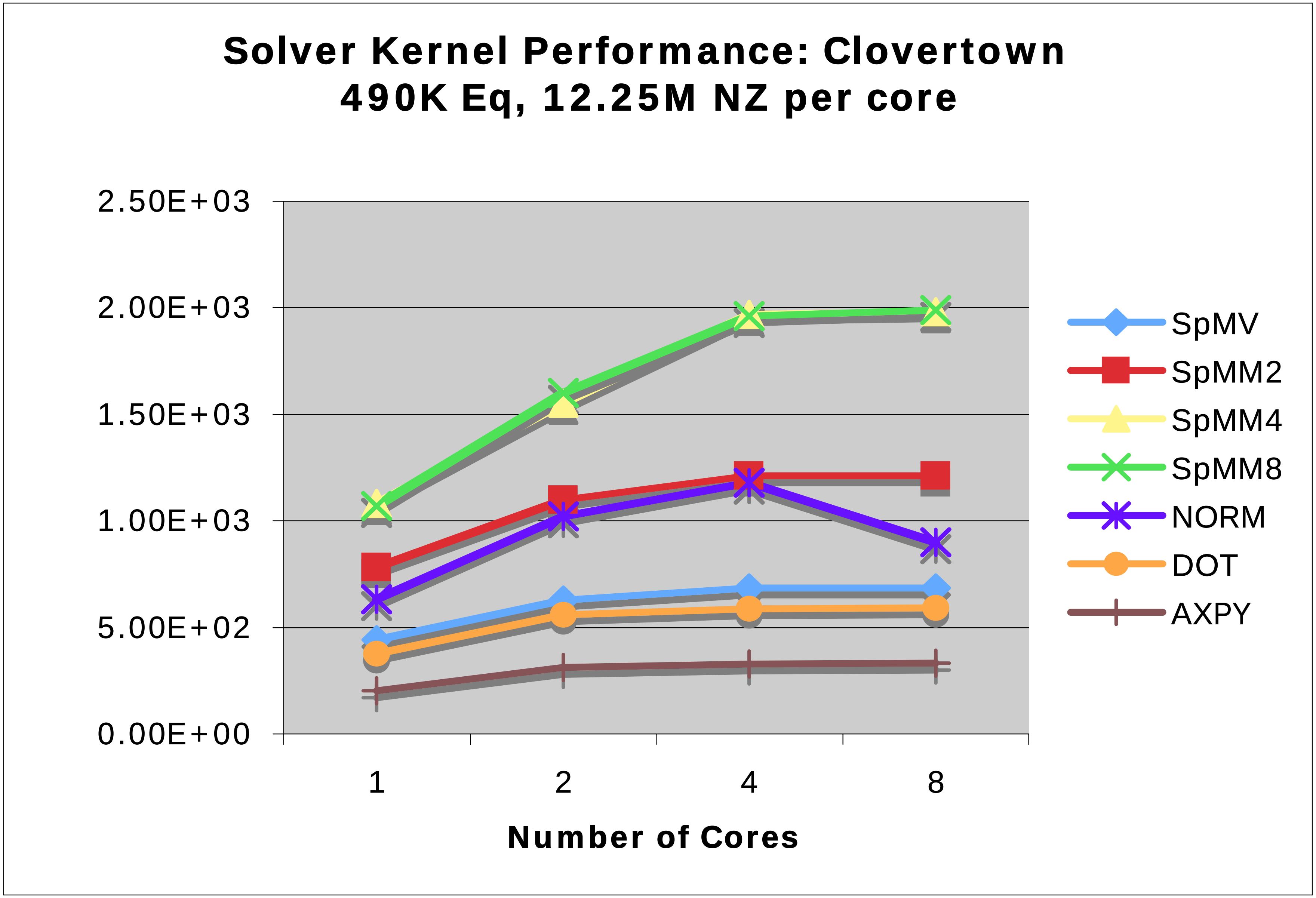
LAMMPS Strong Scaling



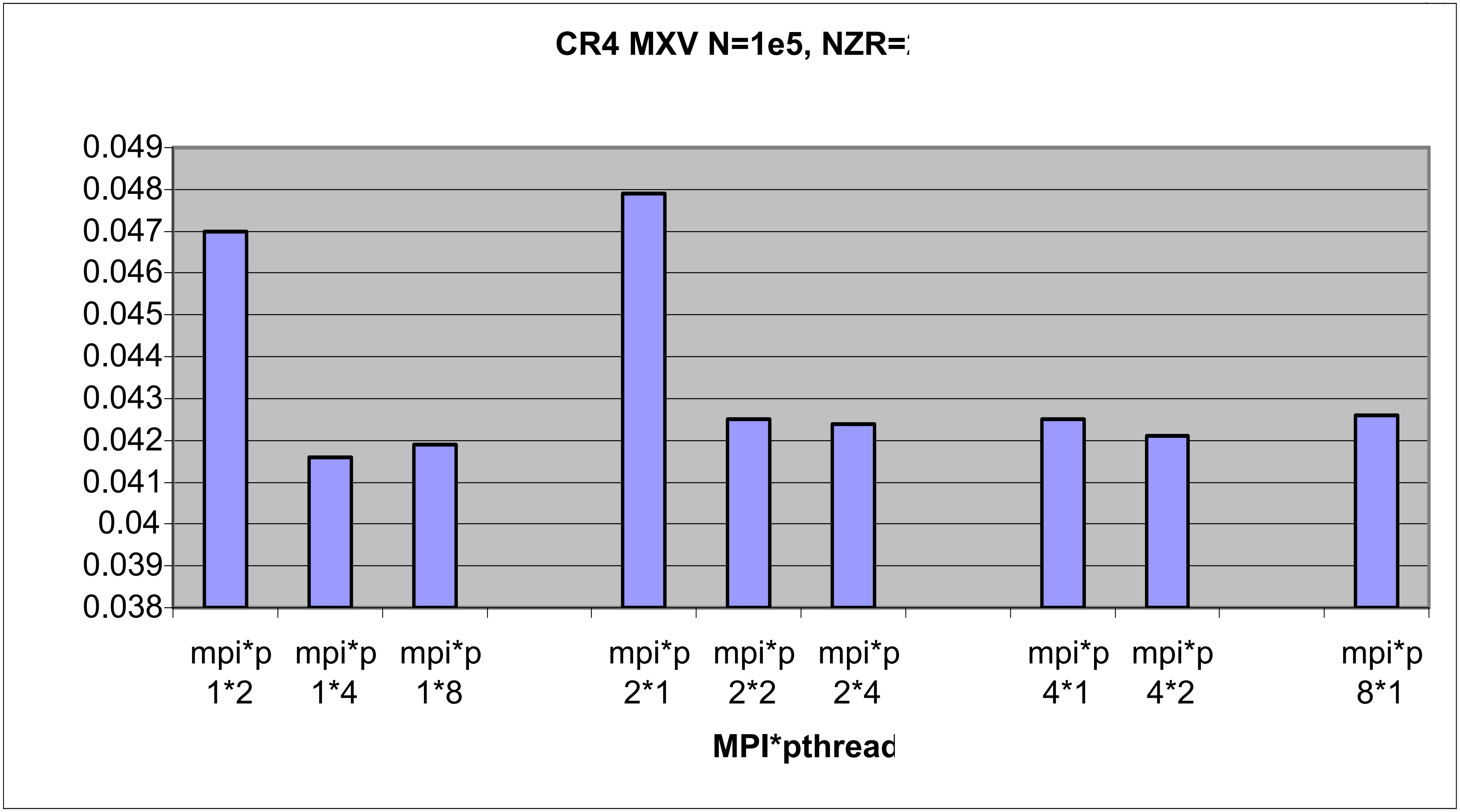
HPC Conjugate Gradient



Sandia National Laboratories

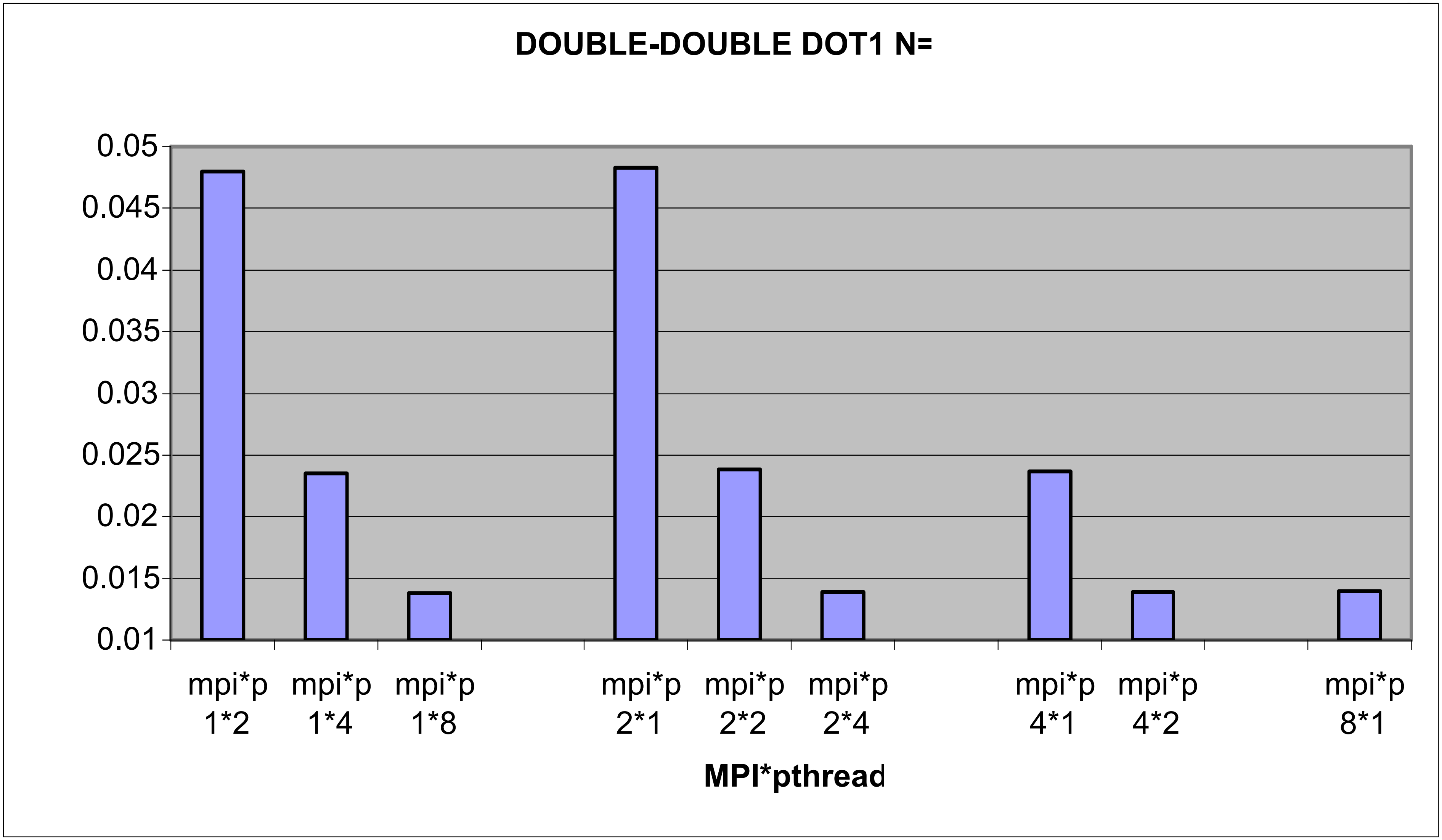


- Trilinos/Epetra MPI Results
- Bandwidth Usage vs. Core Usage

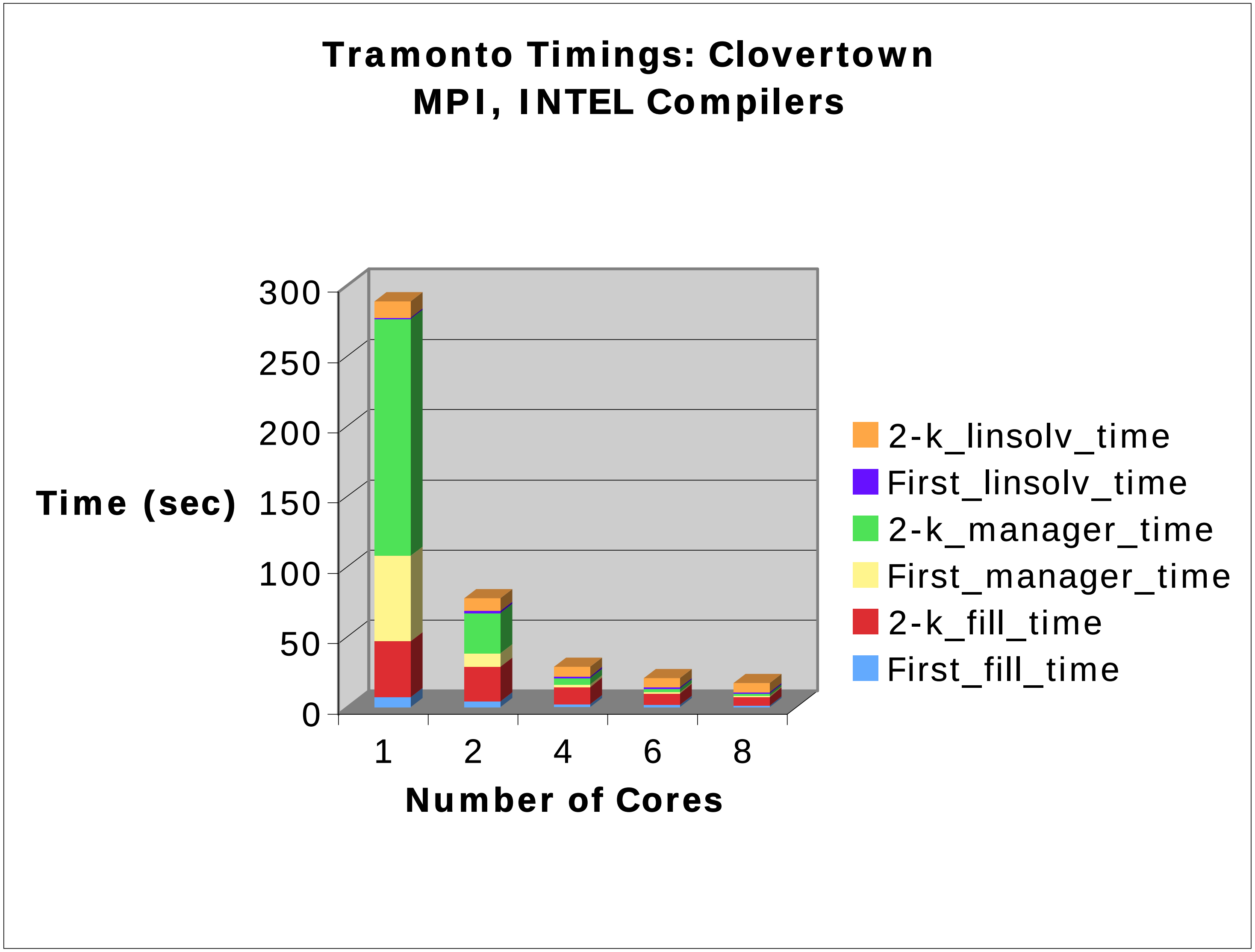


- SpMV MPI+pthreads
- Theme: Programming model doesn’t matter
  if algorithm is the same.

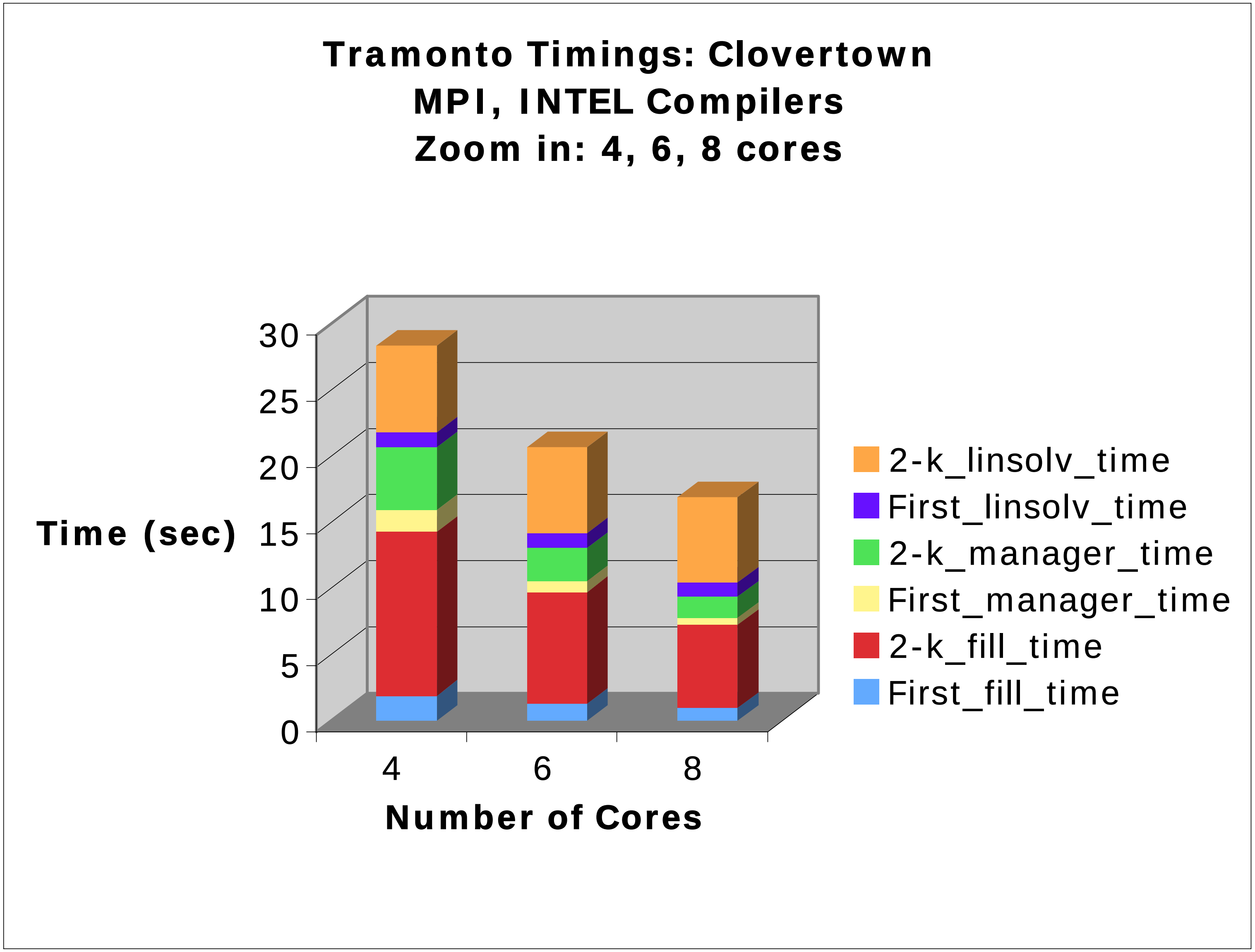
Sandia National Laboratories



- Double-double dot product MPI+pthreads
- Same theme.



- Classical DFT code.
- Parts of code: Speedup is great.
- Parts: Speedup negligible.



- Closer look: 4-8 cores.
- 1 core: Solver is 12.7 of 289 sec (4.4%)
- 8 cores: Solver is 7.5 of 16.8 sec (44%).

Summary: Multicore

- MPI-only is sometimes enough:
  - LAMMPS
  - Tramonto (at least parts), and threads might not help solvers.
- Introducing threads into MPI:
  - Not useful if using same algorithms.
  - Same conclusion as 12 years ago.
- Increase in bandwidth requirements:
  - Decreases effective core use.
  - Independent of programming model.
- Opportunities for effective use of threading:
  - Change of algorithm.
  - Better load balancing.

Sandia National Laboratories

Solver Algorithms for Multicore

- Block Krylov methods: Belos, Anasazi
- Block data structures: VbrMatrix
- Hybrid DMP/SMP preconditioners: Another talk.
- Tpetra focus:
  - Hybrid data strucures.
  - Hybrid parallel machine model.

To Come

Opportunities and Challenges

Themes for FY08/09

- Redefinition of Trilinos scope beyond solvers.
- Next steps in packaging and distribution.
- Continued outreach to other communities
- Rethinking source management.

Scope of Trilinos

- Addition of Sacado, Zoltan, FEI, Intrepid, phdMesh: Not solvers.
- Framework support natural.
- Rephrasing of project goals, descriptions underway.
- Grouping of packages into meta-packages: At least conceptually.

Packaging and Distribution

- Mac and Windows are ever more popular development environments.
- Goal: Provide click-install capabilities for Mac OS, MS Visual Studio, Linux COE.

Outreach

- Trilinos packages part of SciDAC:
  - ITAPS, CSCAPES, TOPS-2.
  - Opportunity to serve broader DOE community.
- Trilinos popular in universities:
  - Single largest sector of users.
- Trilinos part of several industrial efforts.
  - Improves capabilities.
  - Amortizes costs over broader funding sources.
- Elevates certain activities:
  - Fortran accessibility.
  - Packaging & distribution.

Source Management

- Think of repository as a database.
- Logical collections gathered dynamically.
- Consider use of multiple source management tools:
  - Local vs. global management.
  - Fully distributed.
- Certainly svn is option, but looking at all options.

Take Home Messages

- Trilinos is both:
  - A development community
  - A collection of software
- OO techniques lead to:
  - Extensibility at many levels.
  - Scalable infrastructure.
  - Interoperability of independently developed capabilities.
  - Ability to adjust to architecture changes.
- Project is growing:
  - Including more of “vertical software stack”.
  - Adapting to broader user base.
- We are seeking collaborations with broader DOE community.

Sandia National Laboratories

Trilinos Availability/Information

- Trilinos and related packages are available via LGPL.
- Current release (8.0) is "click release". Unlimited availability.
- More information:
  - http://trilinos.sandia.gov
  - http://software.sandia.gov
  - Additional documentation at my website: http://www.cs.sandia.gov/~mheroux.
- 5th Annual Trilinos User Group Meeting:

  November 6-8, 2007 at
  Sandia National Laboratories, Albuquerque, NM, USA.

Sandia National Laboratories