PyTrilinos: A Python Interface to Trilinos, a Set of Object-Oriented Solver Packages

Bill Spotz

Sandia National Laboratories

SciPy 2005

Pasadena, CA 22 Sep 2005

With special thanks to

Marzio Sala, Eric Phipps, Alfred Lorber,

Mike Heroux, Jim Willenbring and Mike Phenow

Sandia National Laboratories

Outline

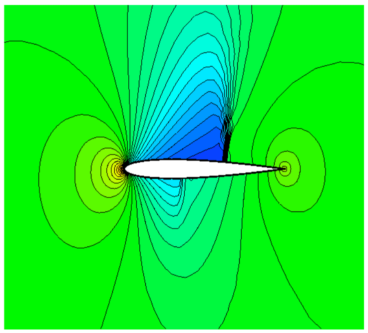
- An Overview of Trilinos
  - Motivation
  - Philosophy & Infrastructure
  - Packages
- An Overview of PyTrilinos
  - Packages
  - Performance
- Summary

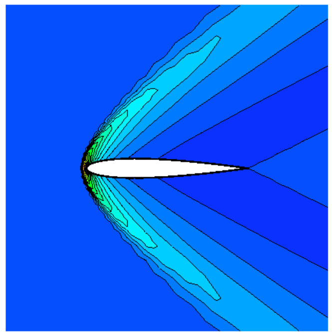
Trilinos Motivation

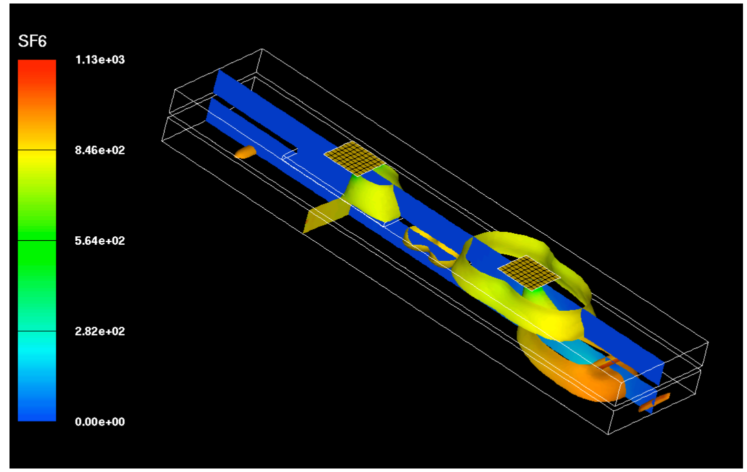
- Sandia does LOTS of solver work
- Challenges
  - Code reuse
  - Leverage development across projects
  - Consistent APIs
  - ASCI SQA/SQE requirements
- Bringing object-oriented tools to scientific computing
  - Frameworks, inheritance, operator overloading…

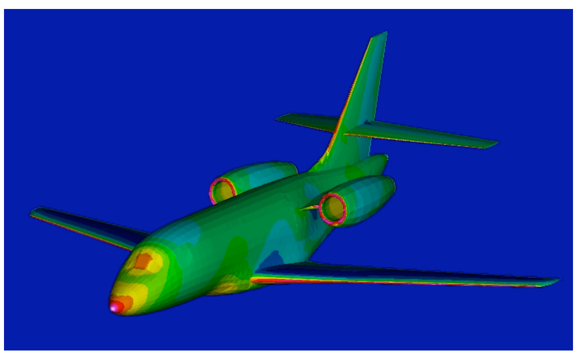
Trilinos Motivation

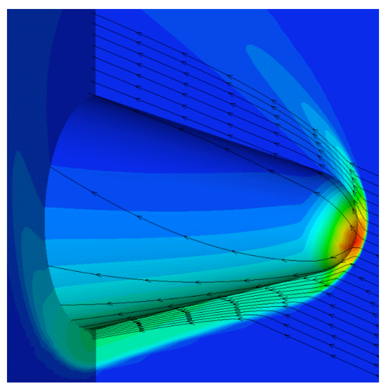
PDEs and Circuits

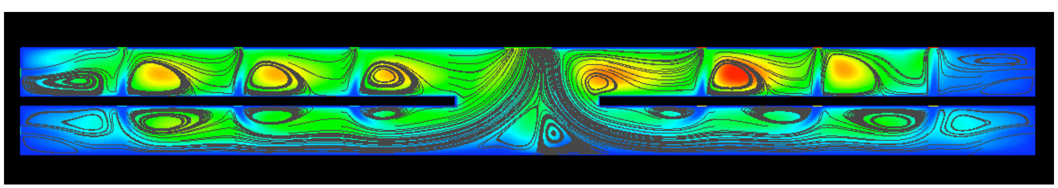


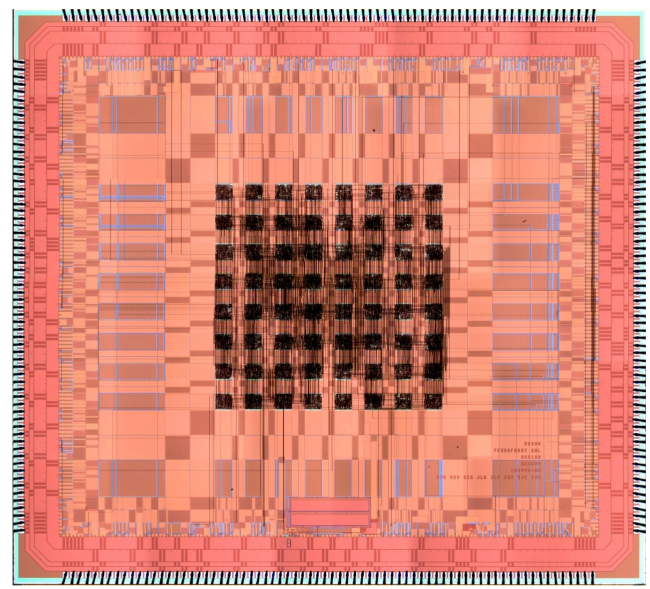












Evolving Trilinos Solution

- **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, bugzilla).
    - Build tools (autotools).
    - Automated regression testing (~20 builds, 5+ platforms, >3000 tests).
    - 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"

Trilinos Development Team

Ross Bartlett
Lead Developer of Thyra
Developer of Rythmos

Paul Boggs
Developer of Thyra

Todd Coffey
Lead Developer of Rythmos

Jason Cross
Developer of Jpetra

David Day
Developer of Komplex

Clark Dohrmann
Developer of CLAPS

Michael Gee
Developer of ML, NOX

Bob Heaphy
Lead developer of Trilinos SQA

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

Ulrich Hetmaniuk
Developer of Anasazi

Robert Hoekstra
Lead Developer of EpetraExt
Developer of Epetra, Thyra, Tpetra

Russell Hooper
Developer of NOX

Vicki Howle
Lead Developer of Meros
Developer of Belos and Thyra

Jonathan Hu
Developer of ML

Sarah Knepper
Developer of Komplex

Tammy Kolda
Lead Developer of NOX

Joe Kotulski
Lead Developer of Pliris

Rich Lehoucq
Developer of Anasazi and Belos

Kevin Long
Lead Developer of Thyra,
Developer of Belos and Teuchos

Roger Pawlowski
Lead Developer of NOX

Michael Phenow
Trilinos Webmaster
Lead Developer of New_Package

Eric Phipps
Developer of LOCA and NOX

Marzio Sala
Lead Developer of Didasko and IFPACK
Developer of ML, Amesos

Andrew Salinger
Lead Developer of LOCA

Paul Sexton
Developer of Epetra and Tpetra

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 and Teuchos

Ray Tuminaro
Lead Developer of ML and Meros

Jim Willenbring
Developer of Epetra and New_Package.
Trilinos library manager

Alan Williams
Developer of Epetra, EpetraExt, AztecOO, Tpetra

Sandia National Laboratories

Trilinos Packages

Linear Algebra Services	Epetra	Kokkos	Komplex	
Linear Solvers	AztecOO	Amesos	Pliris	Belos
Preconditioners	IFPACK	ML	Claps	Meros
Eigensolvers	Anasazi			
Nonlinear Solvers	NOX			
Continuation Algorithms	LOCA			
Abstract Interfaces	Thyra	TSFCore	TSFCoreUtils	TSFExtended
Utilities	Teuchos	EpetraExt	Triutils	Didasko

= Next-Generation

PyTrilinos

National Laboratories

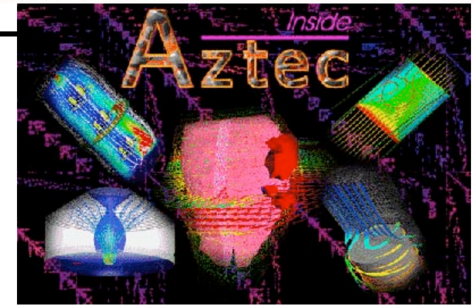
Trilinos Interoperability & Dependence

- Although most Trilinos packages have no explicit dependence, each package must interact with *some* other packages:
  - NOX needs operator, vector and 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 `configure` script is vehicle for:
  - Establishing interoperability of Trilinos components…
  - Without compromising individual package autonomy.

Trilinos Packages: Epetra

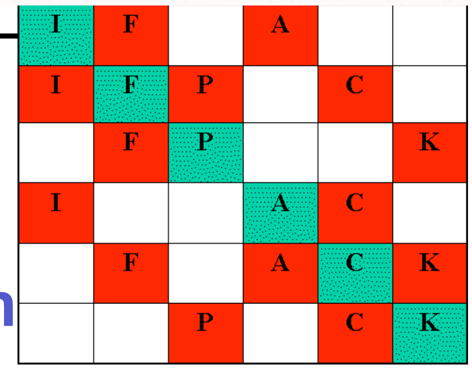
- Petra: “foundation” (E for “essential”)
- Linear Algebra Services
  - Communicators: encapsulate parallelism
  - Maps: describe distribution of LA objects
  - Vectors/multivectors
  - Sparse graphs
  - Sparse matrices
  - Base classes for operators and matrices
  - Views and copies

Trilinos Packages: AztecOO



- Krylov subspace solvers: CG, GMRES, BiCGStab…
- Incomplete factorization preconditioners
- Aztec is the workhorse solver at Sandia
  - Extracted from MPSalsa reacting flow code
  - Dozens of Sandia applications
  - 1900+ external licenses
- AztecOO improves on Aztec by
  - Using Epetra objects
  - Providing more preconditioners/scalings
  - Enabling more sophisticated OO use
- AztecOO interfaces allow:
  - Continued use of Aztec for functionality
  - Introduction of new solver capabilities outside of Aztec

Trilinos Packages: IFPACK



- Algebraic preconditioners
- Overlapping Schwarz preconditioners with **incomplete factorizations**, block relaxations, block direct solves
- Abstract matrix interface (including Epetra)
- Separates graph construction from factorizations
- Compatible with AztecOO, ML, Amesos
- Can be used by NOX and ML

Trilinos Packages: ML

- Multi-level preconditioners
  - Smoothed aggregation
  - Multi-grid
  - Domain decomposition
- Compatibilities:
  - Accepts any implementation of Epetra_RowMatrix
  - Implements Epetra_Operator interface . . . AztecOO
- Can be used completely independent of other Trilinos packages

Trilinos Packages: Amesos

- Distributed sparse direct solvers
- Challenge:
  - Many third-party direct solvers available
  - Different APIs, data formats
  - Interface can change with versions
- Amesos offers:
  - Single, consistent interface
  - Common look and feel for all classes
  - Separation from specific solver details
  - Internal data redistribution
- Third-party packages:
  - LAPACK, KLU, UMFPACK, SuperLU, SuperLU_DIST, MUMPS, ScaLAPACK, DSCPACK, PARDISO, WSMP

Trilinos Packages: NOX

- Suite of nonlinear solution methods
- Uses abstract vector and "group" interfaces:
  - Allows flexible selection and tuning of directions and line searches
  - Abstract vector & group interfaces for Epetra, AztecOO, ML, LAPACK and PETSc
- Controlled by flexible parameter list objects

Trilinos Packages: LOCA

- Library of Continuation Algorithms
- Continuation:
  - Zero-order, first-order, arc length
  - Multi-parameter, turning point, phase transition
  - Pitchfork- and Hopf-bifurcation
- Eigenvalue approximation
  - ARPACK or Anasazi

Trilinos Packages: EpetraExt

- Extensions to Epetra . . . useful, but nonessential
- Examples:
  - Graph/matrix view extraction
  - Zoltan interface
  - Sparse transpose
  - Singleton removal, static condensation filters
  - Overlapped graph constructors
  - Graph coloring algorithms
  - Matlab, MatrixMarket I/O functions
  - Etc…

Trilinos Packages: Anasazi

- Eigensolvers written in templated C++
- Generic interface to a collection of algorithms
- Interfaces are derived from vector and operator base classes

Trilinos Packages: Teuchos

- Utility package of useful tools
- Includes
  - LAPLACK, BLAS wrappers
  - Dense matrix & vector classes
  - FLOP counters, timers
  - Reference-counted pointers
  - Parameter lists
- Uses
  - Templates, STL

Trilinos Packages: Triutils

- Trilinos Utilities (intended for test harness, but sometimes useful elsewhere)
  - **Matrix Galleries**
  - Command-line parser
  - Input file reader

PyTrilinos

- Python interface to selected Trilinos packages
  - Epetra, AztecOO, IFPACK, ML, Amesos, NOX, LOCA, EpetraExt, TriUtils (and New_Package)
- Uses SWIG to generate wrappers
- Prerequisites
  - Python 2.3 or higher
  - Swig 1.3.23 or better
  - Numeric
- Python build system integrated into Trilinos `configure/make` system
  - Building Trilinos is not for the compiler-shy
  - To build PyTrilinos, simply add `--enable-python` (or `--with-python`) to the configure invocation
  - Interfaces will be built for enabled packages w/wrappers
  - `make` calls `swig` and then `setup.py` (`distutils`)
  - My `MakefileVariables` module

Sandia National Laboratories

PyTrilinos.Epetra

```
from PyTrilinos import Epetra  # MPI_Init, MPI_Finalize for MPI builds
comm = Epetra.PyComm()         # Epetra.SerialComm or Epetra.MpiComm
size = 4 * comm.NumProc()      # Scaled problem size
map = Epetra.Map(size,0,comm)  # One of several constructors
v1 = Epetra.Vector(map)        # v1 is also a Numeric array!
print v1
v1.Print()
v1.shape = (2,2)
print v1
```

```
[ 0.  0.  0.  0.]
MyPID          GID              Value
         0             0                       0
         0             1                       0
         0             2                       0
         0             3                       0
[[ 0.  0.]
 [ 0.  0.]]
```

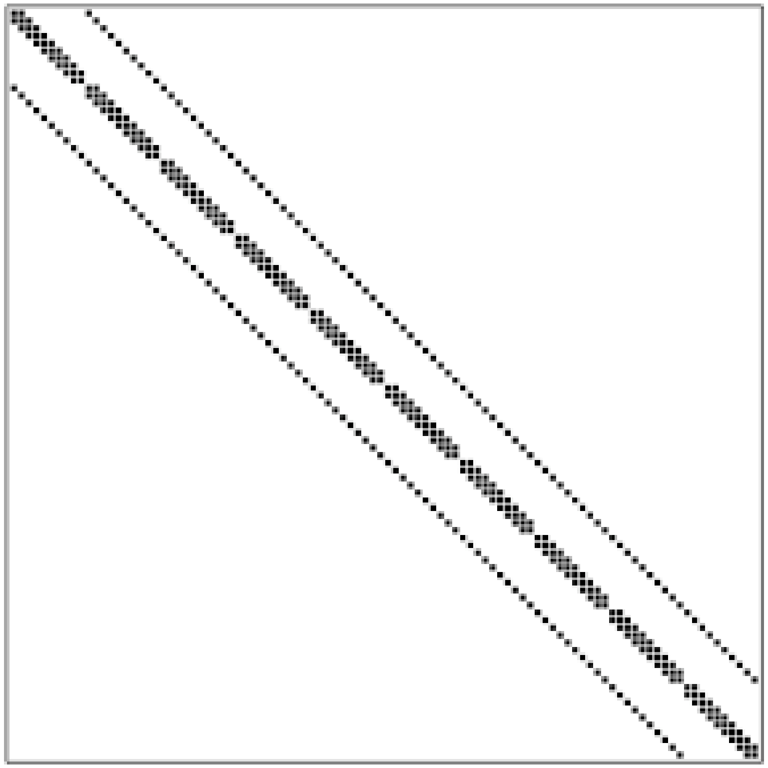
PyTrilinos.Amesos (Triutils, Epetra)

```
from PyTrilinos import Amesos, Triutils, Epetra
comm = Epetra.PyComm()
gallery = Triutils.CrsMatrixGallery("laplace_2d",comm)
gallery.Set("nx",100)
gallery.Set("ny",100)
problem = Epetra.LinearProblem(gallery.GetMatrix(),
                               gallery.GetStartingSolution(),
                               gallery.GetRHS()              )
factory = Amesos.Factory()
solver = factory.Create("SuperLU", problem)
amesosList = {"PrintTiming" : True, "PrintStatus" : True }
solver.SetParameters(amesosList)
solver.SymbolicFactorization()
solver.NumericFactorization()
solver.Solve()
soln = problem.GetLHS()
print "||x_computed||_2 =", soln.Norm2()
```

PyTrilinos.AztecOO (and IFPACK)

```
from PyTrilinos import IFPACK, AztecOO, Tri
comm = Epetra.PyComm()
gallery = Triutils.CrsMatrixGallery("laplac
gallery.Set("nx",8)
gallery.Set("ny",8)
matrix = gallery.GetMatrix(),
lhs    = gallery.GetStartingSolution()
rhs    = gallery.GetRHS()
IFPACK.PrintSparsity(matrix, "matrix.ps")
solver = AztecOO.AztecOO(matrix, lhs, rhs)
solver.SetAztecOption(AztecOO.AZ_solver,
solver.SetAztecOption(AztecOO.AZ_precond,
solver.SetAztecOption(AztecOO.AZ_subdomain_
solver.SetAztecOption(AztecOO.AZ_graph_fill
solver.Iterate(50, 1e-5)        # Max iterat
```

Epetra::CrsMatrix



Sandia National Laboratories

PyTrilinos Performance vs MATLAB

- CPU sec to fill nxn dense matrix

n	MATLAB	PyTrilinos
10	0.00001	0.000416
100	0.0025	0.0357
1000	0.0478	3.857

- CPU sec to fill nxn diagonal matrix

n	MATLAB	PyTrilinos
10	0.00006	0.000159
1000	0.00397	0.0059
10,000	0.449	0.060
50,000	11.05	0.313
100,000	50.98	0.603

- CPU sec for 100 MatVecs

n	MATLAB	PyTrilinos
50	0.02	0.0053
100	0.110	0.0288
500	3.130	1.782
1000	12.720	7.150

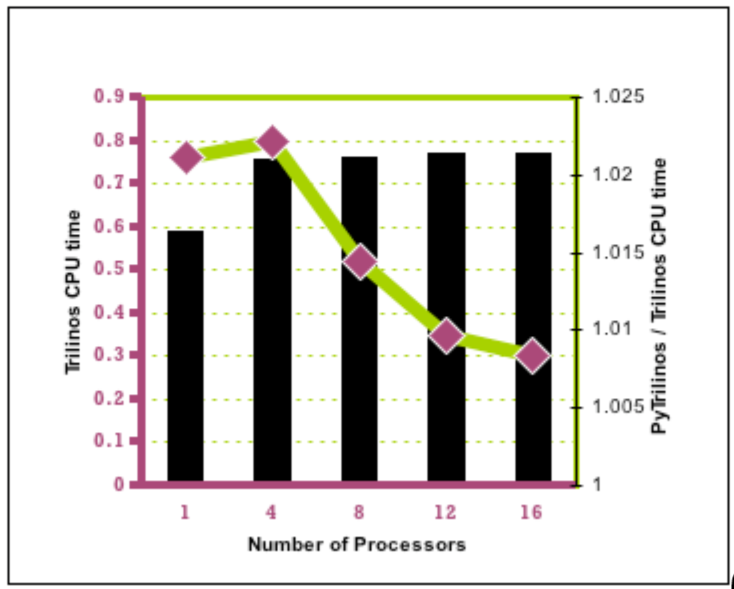
Sandia National Laboratories

PyTrilinos Performance vs Trilinos

- Fine-grained script:

n	Trilinos	PyTrilinos
1000	0.010	0.15
10,000	0.113	0.241
100,000	0.280	1.238
1,000,000	1.925	11.28

- Course-grained script:



Sandia National Laboratories

PyTrilinos Performance

- Some Trilinos packages are designed for users to derive classes from pure virtual base classes
  - Epetra_Operator
  - Epetra_RowMatrix
  - NOX::Abstract::Interface . . .
- Numerical kernels (matvecs, nonlinear function evaluations) are therefore written by users
- Using PyTrilinos, numerical kernels are therefore written in python (fine-grained . . . bad)
- If efficiency is a consideration,
  - Use array slice syntax
  - Use `weave`
  - Inefficient code is 20-100x slower

Summary

- Trilinos is a major software development project at Sandia National Laboratories
  - Interoperable, independent, object-oriented, parallel, sparse linear and nonlinear solver packages
  - Release 6.0: September, 2005
- PyTrilinos provides python access to selected packages
  - Numeric compatibility (NumArray?)
  - Still in early stages . . . portability, guinea pigs
  - Parallelism
  - Rapid prototyping
  - Unit testing
  - Application development