

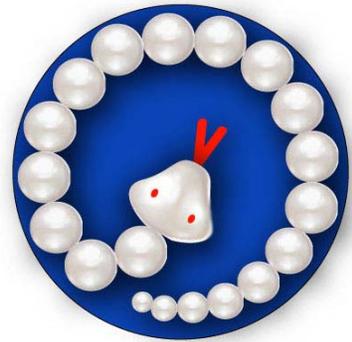
PyTrilinos: A Parallel Python Interface to Trilinos

Bill Spotz

Sandia National Laboratories

**12th SIAM Conference on Parallel
Processing for Scientific Computing
San Francisco, CA 22 Feb 2006**

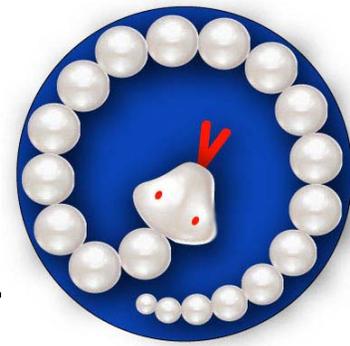
**With special thanks to
Marzio Sala, Eric Phipps, Alfred Lorber,
Mike Heroux, Jim Willenbring and Mike Phenow**



Trilinos Packages

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

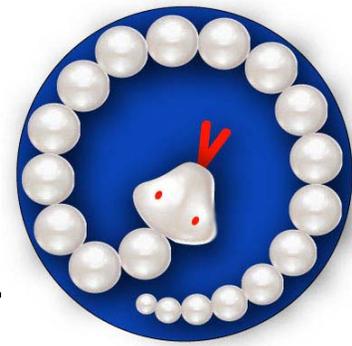
What is PyTrilinos?



- PyTrilinos is a python interface to selected Trilinos packages
- What packages are wrapped?
 - Epetra, EpetraExt, Triutils, Galeri, AztecOO, Amesos, IFPACK, ML, New_Package
 - **Outdated:** NOX, LOCA
 - **Early stages:** Anasazi, Thyra
- Is MPI supported?
 - Yes, it is currently embedded in the Epetra module if Trilinos is configured with `--enable-mpi`

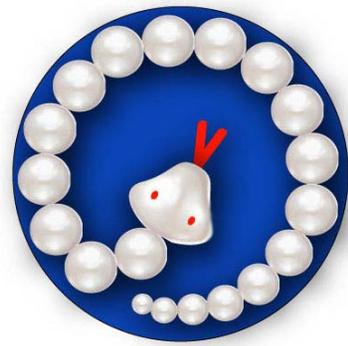


Scripting Interfaces



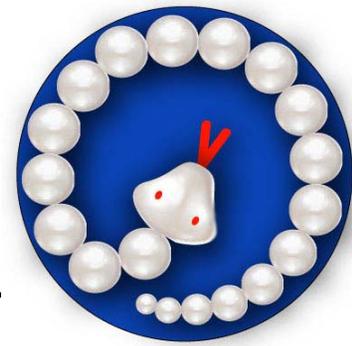
- **Why add a scripting interface to Trilinos?**
 - Interactive creation, manipulation and use of Trilinos objects without compilation step → **rapid prototyping**
 - Application development: scripting languages are good for command-and-control code that can hand off to compiled numerical kernels
- **Why python?**
 - Python was built from the ground up to be object oriented → maps directly to Trilinos design
 - Python was designed to be a teaching language → clean, readable syntax
 - Massive library of standard and third-party modules
 - Large and growing scientific python community

What About SciPy/Numeric (NumPy)?



- SciPy is a huge collection of wrappers for scientific libraries
- Most SciPy packages require multi-dimensional array objects to work on → Numeric (currently migrating to NumPy)
- SciPy's biggest omission is PDE solvers (sparse systems, parallel distributed data, and solvers that can use them)
- PyTrilinos is filling these gaps
- Certain Epetra classes overlap Numeric functionality (e.g. Epetra_MultiVector)
 - Python implementation of these classes inherit from both the Epetra class and Numeric arrays

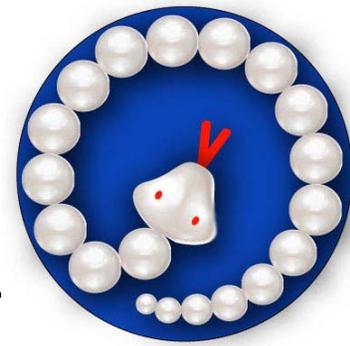
Building & Installing PyTrilinos



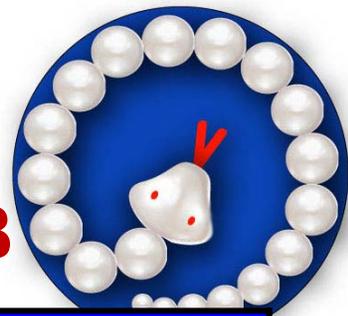
- Prerequisites include python 2.3, Numeric, and swig (Simple Wrapper Interface Generator) 1.3.23
 - Swig is the workhorse for generating wrapper code; wrapper code is not pre-generated because of configuration options
- Add `--enable-python` to invocation of `configure`
- Python modules will be built for those packages that support it



Demonstration



PyTrilinos Performance vs MATLAB



- CPU sec to fill $n \times n$ dense matrix

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

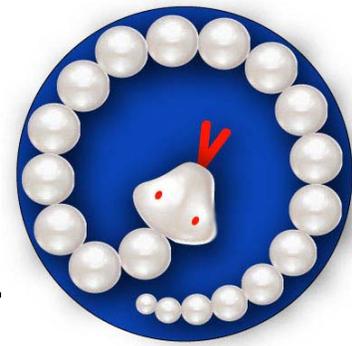
- CPU sec to fill $n \times n$ diagonal matrix

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

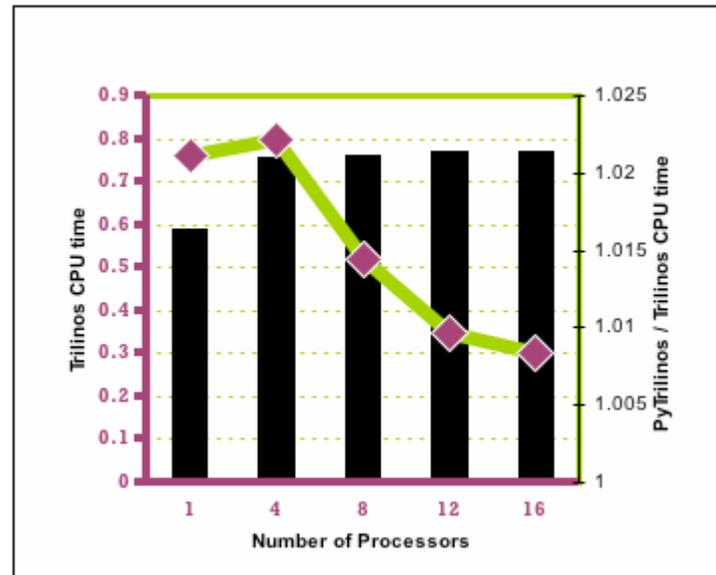
PyTrilinos Performance vs Trilinos



- Fine-grained script:

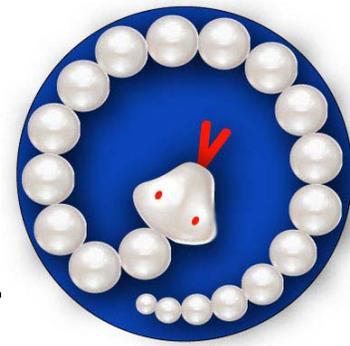
n	Trilinos	PyTrilinos
1000	0.010	0.15
10,000	0.113	0.241
100,000	0.280	1.238
1,000,000	1.925	11.28

- Course-grained script:





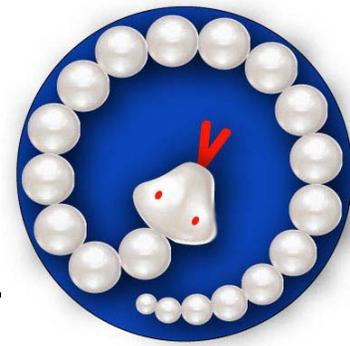
PyTrilinos Performance



- 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



- **PyTrilinos provides python access to selected Trilinos packages**
 - Emerging from early stages . . . portability, completeness
 - Parallelism
 - Rapid prototyping
 - Application development
 - Unit testing
 - Numeric compatibility (migrating to NumPy)
- **PyTrilinos complements and supplements the SciPy package**