PyTrilinos: A Python Interface to Trilinos

Bill Spotz
Sandia National Laboratories

Reproducible Research in Computational Geophysics
August 31, 2006
Computational Sciences at Sandia

- Chemically reacting flows
- Climate modeling
- Combustion
- Compressible flows
- Computational biology
- Electrical modeling
- Heat transfer
- Load balancing

- Materials modeling
- MEMS modeling
- Mesh generation
- Optimization and uncertainty quantification
- Seismic imaging
- Shock and multiphysics
- Structural dynamics
The Trilinos Project

• Provide a central repository for Sandia’s solver technology
• Increase code-reuse
• Organized on concept of “packages”
• Minimize package interdependence
• Maximize package interoperability
• Provide a framework for SQE and SQA
  – Compliance with requirements
  – Nightly test harness
• High degree of developer autonomy
• Open source: GNU Lesser License
• Web site: http://software.sandia.gov/trilinos
• Next release: Version 7.0, September, 2006
• Trilinos Users Group Meeting, November 7-9, 2006
The Trilinos Project

Linear Algebra Services
- Epetra
- Jpetra
- Tpetra
- EpetraExt
- Kokkos
- RTOp

Linear Solvers
- Amesos
- AztecOO
- Belos
- Komplex
- Pliris
- Stratimikos

Nonlinear Solvers
- NOX
- Anasazi
- Rythmos
- LOCA
- MOOCHO

Preconditioners
- IFPACK
- ML
- Claps
- Meros

Meta-Solvers
- Amesos
- AztecOO
- Belos
- Komplex
- Pliris
- Stratimikos

Communicators, maps, vectors, matrices, graphs
- Java
- Sparse kernels
- Templated Reduction, transformation operators
- Extensions

Direct, sparse
- Iterative, sparse
- Iterative, sparse, next generation

Complex interface to AztecOO
- Direct, dense
- Generic interface

Teuchos
- Galeri
- Thyra
- Isorropia
- Moertel
- Didasko
- PyTrilinos
- WebTrilinos
- New_Package

Eigensolvers
- Time-stepping algorithms
- Continuation algorithms
- Constrained optimization

Primary tools package
- Common testing tools
- Matrix gallery
- Common interface

Partitioning & load balance
- Mortar elements
- Trilinos examples

Python Interface
- Web interface

Package template

Incomplete factorization
- Multi-level
- Domain decomposition
- Segregated, block

Sandia National Laboratories
The Interoperability Problem

- Ross has talked about the need for (and work on) a common abstract interface
  - Interfaces between Trilinos packages
  - Interfaces between Trilinos and external packages
- We would like to extend this concept to the python wrappers
  - Interoperability with external python projects, such as SLIMpy
  - Many design decisions still to be made
  - Some early prototyping work has been done
PyTrilinos

- **Linear Algebra Services**
  - **Epetra** (with extensive NumPy compatibility and integration)
  - **EpetraExt** (coloring algorithms and some I/O)
- **Linear Solvers**
  - **Amesos** (LAPACK, KLU, UMFPACK, ScaLAPACK, SuperLU, SuperLUDist, DSCPACK, MUMPS)
  - **AztecOO**
- **Preconditioners**
  - **IFPACK**
  - **ML**
- **Nonlinear Solvers**
  - **NOX** (python wrappers not yet caught up to recent redesigns)
- **Meta-Solvers**
  - **LOCA** (python wrappers not yet caught up to recent redesigns)
  - **Anasazi** (early development stage)
- **Tools and Utilities**
  - **Teuchos** (ParameterList class only)
  - **TriUtils**
  - **Galeri**
  - **Thyra** (early development stage)
  - **New_Package**
Trilinos documentation is handled by doxygen: special comments within code

- Web pages updated twice daily

Python wrappers are generated using swig ... doxygen does not work with swig interface files

- %feature("autodoc", "1");

```python
>>> help(Epetra.Vector.Dot)
```

```
Dot(*args) unbound PyTrilinos.Epetra.Vector method

Dot(self, Epetra_Vector A) -> double
```

Currently working to provide much more extensive documentation highlighting differences between C++ and python interfaces

- Release 7.0 in September
PyTrilinos.Epetra

• Communicators
  – Comm
  – SerialComm
  – MpiComm
  – PyComm

• Maps
  – BlockMap
  – Map
  – LocalMap

• Vectors
  – MultiVector
  – Vector
  – IntVector

• SerialDense objects
  – SerialDenseOperator
  – SerialDenseMatrix
  – SerialDenseVector
  – SerialDenseSolver
  – IntSerialDenseMatrix
  – IntSerialDenseVector

• Graphs
  – CrsGraph

• Operators
  – Operator
  – RowMatrix
  – CrsMatrix
A Quick Detour…

- Python lists are not suitable for scientific computing
  - Flexible but inefficient
  - Heterogeneous data, noncontiguous memory
- NumPy module provides needed functionality
  - Contiguous, homogeneous $n$-dimensional arrays
  - High-level interface
  - Part of SciPy
- SciPy is a large, open source package for a wide variety of python interfaces to scientific software:
  - NetLib’s “greatest hits”
PyTrilinos.Epetra and NumPy

• Array-like classes inherit from `numpy.UserArray`
  – `MultiVector`
  – `Vector`
  – `IntVector`
  – `SerialDenseMatrix`
  – `SerialDenseVector`
  – `IntSerialDenseMatrix`
  – `IntSerialDenseVector`

• Methods throughout Epetra have arguments that accept or produce pointers to C arrays
  – Python input arguments accept python sequences
  – Python output arguments produce `ndarrays`
PyTrilinos.Teuchos

- **Teuchos::ParameterList**
  - Used by several Trilinos packages to set problem parameters
  - Maps string names to arbitrary-type values
  - Python implementation allows dictionary substitutions
  - Hybrid PyDictParameterList objects are returned
  - The following conversions are supported:

<table>
<thead>
<tr>
<th>Python</th>
<th>Dir</th>
<th>C / C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>⇄</td>
<td>bool</td>
</tr>
<tr>
<td>int</td>
<td>⇄</td>
<td>int</td>
</tr>
<tr>
<td>float</td>
<td>⇄</td>
<td>double</td>
</tr>
<tr>
<td>string</td>
<td>⇄</td>
<td>std::string</td>
</tr>
<tr>
<td>string</td>
<td>⇄</td>
<td>char *</td>
</tr>
<tr>
<td>dict</td>
<td>⇒</td>
<td>ParameterList</td>
</tr>
<tr>
<td>wrapped ParameterList</td>
<td>⇄</td>
<td>ParameterList</td>
</tr>
<tr>
<td>wrapped PyDictParameterList</td>
<td>⇒</td>
<td>ParameterList</td>
</tr>
</tbody>
</table>
• Governing equation: \( -\frac{d^2 u}{dx^2} + c \frac{du}{dx} = 0, \quad x \in [0,1] \)

• Boundary conditions: \( u(0) = 0, \quad u(1) = 1 \)

• Exact solution: \( u(x) = \frac{e^{cx} - 1}{e^c - 1} \)

• CDS: \( -\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + c \frac{u_{i+1} - u_{i-1}}{2h} = 0 \)

• Oscillations: \( ch = \frac{c}{n-1} > 2 \)
Conclusions

• **Python lets developers focus on the problem**
  – Memory management, garbage collection
  – Powerful, flexible containers
  – Clean, readable syntax

• **PyTrilinos provides access to powerful solver technologies**
  – Rapid prototyping
  – Application development

• **For computational geophysics, Thyra should provide key too interoperability**
  – Python wrappers for Thyra
  – Thyra adapters for geophysics codes