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
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  – Outdated: NOX, LOCA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
• 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**

<table>
<thead>
<tr>
<th>( n )</th>
<th>MATLAB</th>
<th>PyTrilinos</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.00001</td>
<td>0.000416</td>
</tr>
<tr>
<td>100</td>
<td>0.0025</td>
<td>0.0357</td>
</tr>
<tr>
<td>1000</td>
<td>0.0478</td>
<td>3.857</td>
</tr>
</tbody>
</table>

- **CPU sec to fill \( n \times n \) diagonal matrix**

<table>
<thead>
<tr>
<th>( n )</th>
<th>MATLAB</th>
<th>PyTrilinos</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.00006</td>
<td>0.000159</td>
</tr>
<tr>
<td>1000</td>
<td>0.00397</td>
<td>0.0059</td>
</tr>
<tr>
<td>10,000</td>
<td>0.449</td>
<td>0.060</td>
</tr>
<tr>
<td>50,000</td>
<td>11.05</td>
<td>0.313</td>
</tr>
<tr>
<td>100,000</td>
<td>50.98</td>
<td>0.603</td>
</tr>
</tbody>
</table>

- **CPU sec for 100 MatVecs**

<table>
<thead>
<tr>
<th>( n )</th>
<th>MATLAB</th>
<th>PyTrilinos</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.02</td>
<td>0.0053</td>
</tr>
<tr>
<td>100</td>
<td>0.110</td>
<td>0.0288</td>
</tr>
<tr>
<td>500</td>
<td>3.130</td>
<td>1.782</td>
</tr>
<tr>
<td>1000</td>
<td>12.720</td>
<td>7.150</td>
</tr>
</tbody>
</table>
PyTrilinos Performance vs Trilinos

- Fine-grained script:

<table>
<thead>
<tr>
<th>n</th>
<th>Trilinos</th>
<th>PyTrilinos</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.010</td>
<td>0.15</td>
</tr>
<tr>
<td>10,000</td>
<td>0.113</td>
<td>0.241</td>
</tr>
<tr>
<td>100,000</td>
<td>0.280</td>
<td>1.238</td>
</tr>
<tr>
<td>1,000,000</td>
<td>1.925</td>
<td>11.28</td>
</tr>
</tbody>
</table>

- 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**