An Overview of Trilinos: Packages for Parallel Formulation and Solution of Scientific and Engineering Problems

Michael A. Heroux
Sandia National Laboratories
CScADS Workshop, July-August 2007

Trilinos Contributors

Chris Baker
Developer of Anasazi, RBGen
Ross Bartlett
Lead Developer of MOOCHO, Strdimko, RTO, Thyra
Paul Berg
Developer of Thyra
Erik Boman
Lead Developer of Isomopias
Dennis Boldt
Lead Developer of Aristos
Marino Bova
Lead Developer of Dlibko, Galet, IFPACK, WebTrilinos

Robert Boldt
Lead Developer of EpetraExt
Russell Hooper
Developer of NOX
Vandi Houns
Lead Developer of Merco
Jonathan Hu
Developer of ML
Sarah Knapper
Developer of Kellan
Tammie Kolod
Lead Developer of NOX
Jon Kokoloffi
Lead Developer of Pliris
Rich Lathouer
Developer of Anasazi and Belos
Kevin Lang
Lead Developer of Thyra
Driver of Belos and Taurho
Roger Pawlowski
Lead Developer of NOX
Michael Phaneuf
Trilinos Webmaster
Lead Developer of New Package
Developer of WebTrilinos
Eric Phypas
Lead developer Sacado
Developer of LOCA, NOX

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.
Background/Motivation

Target Problems: PDES and more…

PDES

Inhomogeneous Fluids

Circuits

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

- Desktop: Development and more…
- Capability machines:
  - Redstorm (XT3), Clusters
  - BG/L.
  - 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.
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”

Continued Evolution

- New capabilities extend scope:
  - Sacado: AD Tools. (E. Phipps, D. Gay)
  - Entrep: Discretizations (P. Bochev, D. Ridzal).
  - Others coming…

- Becoming more than just solvers.
Trilinos Statistics

Registered Users by Type (1632 Total)

- University: 925
- Government: 186
- Personal: 175
- Industry: 38
- Other: 30

Registered Users by Region (1632 Total)

- Europe: 579
- US (except Sandia): 219
- Sandia (includes unregistered): 77
- Asia: 208
- Americas (except US): 508
- Australia/NZ: 208
- Africa: 16


External Visibility

- Awards: R&D 100, HPC SW Challenge (04).
- www.cfd-online.com:
  - 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.
- Over 4200 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/

31 July 07
Trilinos Presentation Forums

- ACTS “Hands-on” Tutorial:
  - At Lawrence Berkeley Lab, Berkeley, CA, USA.
- Next Trilinos User Group Meeting:
  - At Sandia National Laboratories, Albuquerque, NM, USA.

TOPS-2 Participation

- Trilinos new member of TOPS.
- Activities start July 2008.
- Looking for application partners.
- Preparation activities:
  - Interoperability with PETSc:
    - PETSc-Trilinos Data objects interoperable.
    - Interoperability at other levels in time.
  - Fortran 9X User Interface:
    - Support construction of matrices/vectors via OO Fortran interface.
    - Support selection of solvers and parameters via Fortran.
  - Attend meetings like this one.
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/documentated 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.
<table>
<thead>
<tr>
<th>Objective</th>
<th>Package(s)</th>
<th>Trilinos Package Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear algebra objects</td>
<td>Epetra, Jpetra, Tpetra</td>
<td>Basic Linear Algebra classes</td>
</tr>
<tr>
<td>Krylov solvers</td>
<td>AztecOO, Belos, Komplex</td>
<td>Block Krylov Methods: Linear: CG, GMRES, Scalable: Krylov-Schur, block Davidson</td>
</tr>
<tr>
<td>ILU-type preconditioners</td>
<td>AztecOO, IFPACK</td>
<td>3rd Party Direct Solver Package.</td>
</tr>
<tr>
<td>Multilevel preconditioners</td>
<td>ML, CLAPS</td>
<td>Abstract Interfaces.</td>
</tr>
<tr>
<td>Eigenvalue problems</td>
<td>Anasazi</td>
<td></td>
</tr>
<tr>
<td>Block preconditioners</td>
<td>Meros</td>
<td></td>
</tr>
<tr>
<td>Direct sparse linear solvers</td>
<td>Amesos</td>
<td></td>
</tr>
<tr>
<td>Direct dense solvers</td>
<td>Epetra, Teuchos, Pliris</td>
<td></td>
</tr>
<tr>
<td>Abstract interfaces</td>
<td>Thyra</td>
<td></td>
</tr>
<tr>
<td>Nonlinear system solvers</td>
<td>NOX, LOCA</td>
<td></td>
</tr>
<tr>
<td>Time Integrators/DAEs</td>
<td>Rythmos</td>
<td></td>
</tr>
<tr>
<td>C++ utilities, (some) I/O</td>
<td>Teuchos, EpetraExt, Kokkos</td>
<td>Full “Vertical” Solver Coverage</td>
</tr>
<tr>
<td>Trilinos Tutorial</td>
<td>Didasko</td>
<td></td>
</tr>
<tr>
<td>“Skins”</td>
<td>PyTrilinos, WebTrilinos, Star-P, Stratimikos</td>
<td></td>
</tr>
<tr>
<td>Optimization</td>
<td>MOOCHO, Aristos</td>
<td></td>
</tr>
<tr>
<td>Archetype package</td>
<td>NewPackage</td>
<td></td>
</tr>
<tr>
<td>Others</td>
<td>Galeri, Isorropia, Moertel, RTOp, Sacado</td>
<td></td>
</tr>
</tbody>
</table>

### Full “Vertical” Solver Coverage

<table>
<thead>
<tr>
<th>Trilinos Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOOCHO</td>
</tr>
<tr>
<td>Rythmos</td>
</tr>
<tr>
<td>NOX</td>
</tr>
<tr>
<td>LOCA</td>
</tr>
<tr>
<td>AztecOO, Belos, Ifpack, ML, etc.</td>
</tr>
<tr>
<td>Anasazi</td>
</tr>
<tr>
<td>Epetra, Tpetra</td>
</tr>
</tbody>
</table>
New in Trilinos 8.0

- Sacado (E. Phipps, D. Gay) AD tools via templated types:
  - Write residual evaluation using template scalar type.
  - Get Jacobian “for free.”
- Belos (H. Thornquist, M. Parks) Block Krylov Linear solvers:
  - Block CG.
  - Block GMRES:
    - Modular implementation: e.g., orthogonalization interface, several adapters.
    - All BLAS 3 kernels.
  - Pseudo-Block GMRES:
    - Synch’ed independent GMRES’s.
    - BLAS 3 SpMM, SpSM.
  - Single RHS GCRO-DR: Recycling GMRES:
    - Useful for sequences of systems: \( A'x' = b', \ i = 1, \ldots \)
- Aristos (D. Ridzal) Nonlinear continuous optimization package:
  - Based on full-space SQP methods.
  - Specifically designed for large-scale constrained optimization problems with inexact linearized constraint equations solves.

Managing Parallel Data

The Petra Object Model
A Simple Epetra/AztecOO Program

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

    // ***** Create x and b vectors *****
    Epetra_Vector x(Map);
    Epetra_Vector b(Map);
    b.Random(); // Fill RHS with random #s

    // ***** 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 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;
}
```

Typical Flow of Epetra Object Construction

- **Construct Comm**
  - Any number of Comm objects can exist.
  - Comms can be nested (e.g., serial within MPI).

- **Construct Map**
  - Maps describe parallel layout.
  - Maps typically associated with more than one comp object.
  - Two maps (source and target) define an export/import object.

- **Construct x**
- **Construct b**
- **Construct A**
  - Computational objects.
  - Compatibility assured via common map.
Perform redistribution of distributed objects:
• Parallel permutations.
• “Ghosting” of values for local computations.
• Collection of partial results from remote processors.

Petra Object Model
Base Class for All Distributed Objects:
• Performs all communication.
• Requires Check, Pack, Unpack methods from derived class.

Graph class for structure-only computations:
• Reusable matrix structure.
• Pattern-based preconditioners.
• Pattern-based load balancing tools.
• Redistribution of matrices, vectors, etc.

Describes layout of distributed objects:
• Vectors: Number of vector entries on each processor and global ID
• Matrices/graphs: Rows/Columns managed by a processor.
• Called “Maps” in Epetra.

Dense Distributed Vector and Matrices:
• Simple local data structure.
• BLAS-able, LAPACK-able.
• Ghostable, redistributable.
• RTOp-able.

Petra Implementations

• Three version under development:
  • 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.
Epeters Performance Optimization Guide
SAND2005-1668

- Topics:
  - 3rd Party Libraries: BLAS and LAPACK
  - Epetera_MultiVector Data Layout
  - Epetera_CrsGraph Construction
  - Epetera_CrsMatrix Construction
  - Selecting the Right Sparse Matrix Class
  - Parallel Data Redistribution
  - General Practices
- Tiered Approach to practices.

Practice Categories

Each practice falls into one of three categories:

- **Very Strongly Recommended** - Practices necessary for Epetera to perform well.
- **Strongly Recommended** - Practices that are definitely a good thing or that have proved to be valuable.
- **Recommended** - Practices that are probably a good idea.
Data Model Wrap-Up

- Our target apps and computer platforms require flexible data model.
- Petra Object Model (POM) supports:
  - Arbitrary placement of vector, graph, matrix entries on parallel machine.
  - Arbitrary redistribution, ghosting and collection of distributed data.
  - Coarse-grain abstraction for key objects.
- This flexibility is needed by LALs:
  - Algebraic and multi-level preconditioners.
  - Concrete distributed matrix kernels.
  - Direct methods.
- Also needed for future architectures: Multicore, GPUs, CELL, etc.
- Also needed for future parallel languages & libraries: UPC, CAF, etc.
- Final Notes:
  - POM is complex: BUT Non-LALs (ANAs) do not rely on it.
  - Parallel Data Redistribution is a huge topic I must skip for this talk (ask about it at break if interested).

31-Jul-07

Templates and Generic Programming

31-Jul-07
Investment in Generic Programming

- 2nd Generation Trilinos packages are templated on:
  - OrdinalType (think int).
  - ScalarType (think double).
- Examples:
  - Teuchos::SerialDenseMatrix<int, double> A;
  - Teuchos::SerialDenseMatrix<short, float> B;
- Scalar/Ordinal Traits mechanism completes support for genericity.
- The following packages support templates:
  - Teuchos (Basic Tools)
  - Thyra (Abstract Interfaces)
  - Tpetra (including MPI support)
  - Belos (Krylov and Block Krylov Linear),
  - IFPACK (algebraic preconditioners, next version),
  - Anasazi (Eigensolvers),
  - Sacado (AD),
  - MOOCHO(Optimization)

Potential Benefits of Templated Types

Templated scalar (and ordinal) types have great potential:
- Generic: Algorithms expressed over abstract field.
- High Performance: Partial template specialization + Fortran.
- Facilitate variety of algorithmic studies.
- Allow study of asymptotic behavior of discretizations.
- Facilitate debugging: Reduces FP error as source error.
- UQ, QMU studies.
- Use your imagination…
- All new Trilinos packages are templated.
- Dark side: Compilation times can soar (but don’t have to).
Other Interface Packages: “Skins”

*PyTrilinos, WebTrilinos, ForTrilinos*

---

### PyTrilinos

- PyTrilinos provides Python access to Trilinos packages.
- Uses SWIG to generate bindings.
- Epetra, AztecOO, Galeri, IFPACK, ML, NOX, LOCA, Teuchos, Thyra, Anasazi, 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.
```cpp
#include "mpi.h"
#include "Epetra_MpiComm.h"
#include "Epetra_Time.h"
#include "Epetra_RowMatrix.h"
#include "Epetra_Time.h"
#include "Trilinos_Util_CrsMatrixGallery.h"
using namespace Trilinos_Util;

int main(int argc, char *argv[]) {
  MPI_Init(&argc, &argv);
  Epetra_MpiComm Comm(MPI_COMM_WORLD);
  int nx = 1000;
  int ny = 1000 * Comm.NumProc();
  CrsMatrixGallery Gallery("laplace_2d", Comm);
  Gallery.Set("nx", nx);
  Gallery.Set("ny", ny);
  Gallery.Set("problem_size", nx*ny);
  Gallery.Set("map_type", "linear");
  Epetra_LinearProblem* Problem = Gallery.GetLinearProblem();
  assert (Problem != 0);
  Comm = Epetra.PyComm()

  // retrieve pointers to solution (lhs), right-hand side (rhs) and matrix itself (A)
  Epetra_MultiVector* rhs = Problem->GetLHS();
  Epetra_MultiVector* rhs = Problem->GetRHS();
  Epetra_RowMatrix* A = Problem->GetMatrix();
  Epetra_Time Time(Comm);
  for (int i = 0 ; i < 10 ; ++i)
    A->Multiply(0, *lhs, *rhs);
  cout << Time.ElapsedTime() << endl;
  MPI_Finalize();
  return(EXIT_SUCCESS);
} // end of main()
```

---

**Web Interface**

- Yet another Trilinos Package
- Supports:
  - Use of Trilinos via simple Web Editor
  - User or generated data sets.
  - Compilation and execution on webserver.

---

**webtrilinos**

**Welcome to WebTrilinos!**

This is the default home page of your personal installation of WebTrilinos. First, you have to fix a few things, and specify a few instructions.

- Check the basic requirements.
- Go to the setup page.
- Check the settings.

**Important note:** Remember to add a password protection to the directories `trilinos/cpp` and `trilinos/python/installdir`.

At this point you can use your installation of WebTrilinos. Currently, WebTrilinos offers the following three modules:

1. **C++/Python interface:** A development environment for testing basic Trilinos programs, having to install any Trilinos packages are included.
2. **C++/Python interface:** For the C++ interface, but for writing Python. Examples of usage of PyTrilinos are included.

For more details, please check the official `WebTrilinos` webpage.
Sample Page

- Intended use:
  - Installation on departmental intranet.
  - Allows first time users easy access.
  - Supports “just-in-time” use of solvers.

Fortran Interface

- Presently Trilinos has no full-featured Fortran interface.
- Plans in place to develop OO Fortran 9X interface.
- Will Develop as part of SciDAC TOPS-2 effort.
- Interested in a target Fortran application for collaboration.
Formal Software Engineering

CSE Software is not IT Software

---

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

<table>
<thead>
<tr>
<th>Trilinos Service</th>
<th>SQE Practices Impact</th>
</tr>
</thead>
</table>
| **Yearly Trilinos User Group Meeting (TUG) and Developer Forum:**                | — All Requirements steps: gathering, derivation, documentation, feasibility, etc.  
| Once a year gathering for tutorials, package feature updates, user/developer requirements discussion and developer training. | — User and Developer training.  
| **Monthly Trilinos leaders meetings:**                                          | — Developer Training.  
| 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. | — Design reviews.  
| — All Requirements steps: gathering, derivation, documentation, feasibility, etc. | — Policy decisions across all development phases.  
| **Trilinos and package mail lists:**                                            | — Developer/user/client communication.  
| Trilinos lists for leaders, announcements, developers, users, checks and similar lists at the package level support a variety of communication. All lists are archived, providing critical artifacts for assessments and audits. | — Requirements/design/testing artifacts.  
| — User and Developer training.  
| **Trilinos and Trilinos3PL source repositories:**                               | — Announcement/documenting of releases.  
| 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.  
| — Design reviews.  
| **Bugzilla Products:**                                                           | — Third-party software management.  
| Each package has its own Bugzilla Product with standard components.              | — Requirements/faults capturing and tracking.  
| **Trilinos configure script and M4 macros:**                                    | — Portability.  
| The Trilinos configure script and related macros support portable installation of Trilinos and its packages | — Software release.  
| **Trilinos test harness:**                                                       | — Pre-checkin and regression testing.  
| 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. | — Software metrics.  

### “Promotional” Model

- **Phase k**
  - Lower formality
  - Fewer Artifacts
  - Lean/Agile

- **Promotional Event**

- **Phase k+1**
  - Higher formality
  - Sufficient Artifacts
  - Bullet proof
  - Maintainable

31 July 07
Trilinos Software Lifecycle Model

- 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.


A Few More Useful Things
**Stratimikos**

- New package in Trilinos 7.0.
- Single point of access to Trilinos preconditioners/solvers:
  - Uniform interface to all preconditioners.
  - Uniform interface to all solvers, linear and non-linear.
  - Selection of preconditioner/solver via parameter list.
- Simplest way to access the suite of Trilinos capabilities
- Simple driver code available on website.

**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.
The Trilinos Tutorial Package

- Trilinos is large (and still growing…)
  - More than 900K code lines
  - Many packages
    - a lot of functionalities…
    - … and also a lot to learn

- Trilinos tutorial is also delivered as a package: Didasko.

What DIDASKO is not

- DIDASKO, as a tutorial, cannot cover the most advanced features of Trilinos:
  - Only the “stable” features are covered

- DIDASKO is not a substitute of each package’s documentation and examples, which remain of fundamental importance
Covered Packages

- At present, we cover

<table>
<thead>
<tr>
<th>Package</th>
<th>Triutils</th>
<th>IFPACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epetra</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teuchos</td>
<td>AztecOO</td>
<td>ML</td>
</tr>
<tr>
<td>NOX</td>
<td>Amesos</td>
<td>EpetraExt</td>
</tr>
<tr>
<td>Anasazi</td>
<td>Tpetra</td>
<td></td>
</tr>
</tbody>
</table>

Some Upcoming Efforts

- Exploiting new architecture opportunities:
  - Multicore especially.
- Redefining the scope of Trilinos beyond solvers:
  - CSE enabling technologies distribution.
- Expanding accessibility:
  - Fortran Interfaces.
  - Easier installation (esp. on Windows).
Challenges

- Support and perfective maintenance of existing software:
  - Presently done through growth, changes in system architecture.
  - Not sustainable long term.
- Adaptation of software engineering research to CSE software.
  - Imposition of standard IT approach worse than ineffective.
  - Must have counter-proposals of our own.

Trilinos Availability/Information

- Trilinos and related packages are available via LGPL.
- Current release (7.0) is “click release”. Unlimited availability.
- More information:
- 5th Annual Trilinos User Group Meeting: November 6-8, 2007 at Sandia National Laboratories, Albuquerque, NM, USA.
Summary

- Trilinos is a large, growing, modular software delivery framework:
  - Some initial investment in understanding design is typical
    (especially if unfamiliar with C++, configure/make, OO principles)
  - Modularity allows focusing on your scope of interest.
- Interested in new collaborations.