Trilinos Progress, Challenges and Future Plans

Michael A. Heroux
Sandia National Laboratories
THREAD-SCALEABLE PROGRAMMING KOKKOS
Kokkos Key Features

- Multi-dimensional array containers:
  - Index space abstraction:
    - Physics (i,j,k) does not dictate storage (i,j,k).
  - Physical layout determined by:
    - Memory placement policy.
    - At compile time.

- Parallel patterns execution framework:
  - parallel_for/reduce/scan – Similar to TBB.
  - Task-DAG – under development.

- Goals:
  - Provide portability layer for apps and libraries.
  - Provide prototypes for eventual C++ standards.
Kokkos: A Layered Collection of Libraries

- Applications and Domain Libraries written in Standard C++
  - *Not* a language extension like OpenMP, OpenACC, OpenCL, CUDA, ...
  - Require C++1998 standard (supported everywhere except IBM’s xlC)
  - Prefer C++2011 for its concise lambda syntax (LLNL’s RAJA requires this)
    - As soon as vendors catch up to C++2011 language compliance

- Kokkos implemented with C++ template meta-programming
  - In *spirit* of TBB, Thrust & CUSP, C++AMP, LLNL’s RAJA, ...

Application and Domain Specific Library Layer

- Kokkos Sparse Linear Algebra
- Kokkos Containers
- Kokkos Core

Back-ends: Cuda, OpenMP, pthreads, vendor libraries ...
Porting in Progress: Trilinos

- Trilinos: SNL’s suite of equation solver libraries (and others)
  - Currently MPI-only parallel
  - Incremental refactoring to MPI+Kokkos parallel
- Tpetra: Trilinos’ core parallel sparse linear algebra library
  - Vectors, multi-vectors, sparse matrices, parallel data distribution maps
  - Fundamental operations: axpy, dot, matrix-vector multiply, ...
  - Templated on “scalar” type: float, double, automatic differentiation (AD), embedded uncertainty quantification (UQ), ...
- Port Tpetra to MPI+Kokkos, other libraries follow
  - On schedule to complete in Spring 2015
  - Use of NVIDIA’s unified virtual memory (UVM) expedited porting effort
- Embedded UQ already Kokkos-enabled through LDRD
  - Greater computational intensity leads to significant speed-ups compared to non-embedded UQ sampling algorithms
Node API Futures: MPI Analogy

- `ls Trilinos/packages/aztecoo/src/md_wrap_*`
  
  `md_wrap_intel_c.c  md_wrap_ncube_c.c  md_wrap_scalar_c.c  
  md_wrap_mpi_c.c  md_wrap_puma_c.c  md_wrap_sp2_c.c`

- Excerpt from `md_wrap_sp2_c.c` (next slide).
int md_write(char *buf, int bytes, int dest, int type, int *flag)
{
    int err, buffer;

    if (bytes == 0) {
        err = mpc_bsend(&buffer, 1, dest, type);
    } else {
        err = mpc_bsend(buf, bytes, dest, type);
    }
    if (err!=0) (void) fprintf(stderr, "mpc_bsend error = %d\n", mperrno);
    return 0;
}
int md_write(char *buf, int bytes, int dest, int type, int *flag)
{
#if defined (MPL)
    int err, buffer;

    if (bytes == 0) {
        err = mpc_bsend(&buffer, 1, dest, type);
    } else {
        err = mpc_bsend(buf, bytes, dest, type);
    }
    if (err!=0) (void) fprintf(stderr, "mpc_bsend error = %d\n", mperrno);
#else defined (MPI)
    int err, buffer;
    if (bytes == 0) {
        err = MPI_Send(&buffer, 1, MPI_BYTE, dest, type, MPI_COMM_WORLD);
    } else {
        err = MPI_Send(buf, bytes, MPI_BYTE, dest, type, MPI_COMM_WORLD);
    }
    if (err != 0) (void) fprintf(stderr, "MPI_Send error = %d\n", err);
#endif
    return 0;
}
All node APIs are transitional…

- Except the one(s) that becomes the standard.
- Recommended activities:
  - Write a light-weight API:
    - Compile-time polymorphism.
    - Match your needs and nothing more.
    - Wrap other functionality: CUDA, OpenMP, OpenCL, pthreads, …
    - Don’t fall in love with your implementation!
    - Examples: OCCA, Mint, Kokkos, RAJA, and more.
  - Participate in standards committees.
  - Prepare for eventual replacement by a standard.
TRILINOS EMBEDDED NONLINEAR ANALYSIS TOOLS
Embedded Nonlinear Analysis Tools

- solution of nonlinear equations
- time integration
- bifurcation tracking / stability analysis / parameter continuation
- optimization (black-box, PDE-constrained, full-space)
- uncertainty quantification
- multi-physics coupling
- model order reduction

Governing Philosophy: “Analysis beyond Simulation,”
Goal: to automate many computational analysis and design tasks, using applied math and algorithms to replace trial-and-error or repeated simulation.

- parameter studies
- sensitivity analysis
- calibration
- optimization
- locating instabilities
- performing UQ

Align with Trilinos Strategic Goals:
1. Full Vertical Coverage
2. Hardened Solvers
3. Scalability
<table>
<thead>
<tr>
<th>Package Name</th>
<th>Quick Description</th>
<th>Point of Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piro</td>
<td>Uniform Wrapper for most ENAT Capabilities</td>
<td>Andy Salinger</td>
</tr>
<tr>
<td>NOX</td>
<td>Nonlinear Solver with Globalized Newton’s methods</td>
<td>Roger Pawlowski</td>
</tr>
<tr>
<td>LOCA</td>
<td>Parameter Continuation, Bifurcation Tracking, 4D</td>
<td>Eric Phipps</td>
</tr>
<tr>
<td>Rythmos</td>
<td>Time integration algorithms</td>
<td>Eric Phipps</td>
</tr>
<tr>
<td>Sacado</td>
<td>Automatic Differentiation using Expression Templates</td>
<td>Eric Phipps</td>
</tr>
<tr>
<td>Stokhos</td>
<td>Stochastic-Galerkin Uncertainty Quantification Tools</td>
<td>Andy Salinger</td>
</tr>
<tr>
<td>TriKota</td>
<td>Interface to Dakota for a Trilinos app</td>
<td>Ridzal, Kouri</td>
</tr>
<tr>
<td>PIKE</td>
<td>Multi-Physics coupling</td>
<td>Pawlowski</td>
</tr>
<tr>
<td>Moocho</td>
<td>Embedded (PDE-constrained) Optimization, rSQP</td>
<td>Roscoe Bartlett</td>
</tr>
<tr>
<td>OptiPack</td>
<td>Nonlinear CG</td>
<td>Roscoe Bartlett</td>
</tr>
<tr>
<td>GlobiPack</td>
<td>Library of Line search methods for globalizations</td>
<td>Roscoe Bartlett</td>
</tr>
<tr>
<td>Aristos</td>
<td>Full-Space Optimization</td>
<td>Denis Ridzal</td>
</tr>
<tr>
<td>Razor</td>
<td>Model Order Reduction</td>
<td>Cortial &amp; Carlberg, Kalashnikova</td>
</tr>
</tbody>
</table>

**Related Efforts Outside of Trilinos**

- **Dakota**: Dakota is a mature and widely-used software toolkit that delivers many analysis capabilities using a non-intrusive (a.k.a. blackbox) interface...
- **Albany, Panzer**: Codes built in Albany or on Panzer are born with transformational analysis capabilities.
Embedded Nonlinear Analysis Tools and Key Abstract Interfaces

- Piro
  - Analysis Tools *(black-box)*
    - Dakota
    - ROL
    - OptiPack

- Analysis Tools *(embedded)*
  - NOX
  - Rythmos
  - LOCA
  - LOCA
  - Stokhos
  - Anasazi
  - ROL

- Application Assembly
  - Discretizations

- Derivative Tools
  - Sacado
  - Stokhos

- ModelEvaluator

- Composite Physics
  - PIKE
    - Nonlinear Elim.

- ROL
- Adjoint
- PIKE
- Stokhos

- ROL Ridzal Wed 10:00
- Adjoint Perego Wed 10:50
- PIKE Pawlowski Wed 11:10
- Stokhos Phipps Wed 2:40
RESILIENT COMPUTING & TRILINOS
Four Resilient Programming Models

- Relaxed Bulk Synchronous (rBSP)
- Skeptical Programming. (SP)
- Local-Failure, Local-Recovery (LFLR)
- Selective (Un)reliability (SU/R)

Toward Resilient Algorithms and Applications
Michael A. Heroux arXiv:1402.3809v2 [cs.MS]
rBSP: Reducing synchronization costs “Underlapping” Domain Decomposition


(a) Adjacency Graph of Local Submatrix.  (b) Local Submatrix
Speed up by reducing latency sensitivity
Skeptical Programming

I might not have a reliable digital machine

- Expect rare faulty computations
- Use analysis to derive cheap “detectors” to filter large errors
- Use numerical methods that can absorb bounded error

**Algorithm 1: GMRES algorithm**

```
for l = 1 to do
    r := b - Ax
    q1 := r / ||r||
    for j = 1 to restart do
        w0 := Aqj
        for i = 1 to j do
            hi,j := qi · wi-1
            wi := wi-1 - hi,j qi
        end
        hj+1,j := ||w||
        qj+1 := w / hj+1,j
        Find y = min ||Hjy - b||
        Evaluate convergence criteria
        Optionally, compute xj = Qjy
    end
end
```

**GMRES**

**Theoretical Bounds on the Arnoldi Process**

\[ \|w_0\| = \|Aq_j\| \leq \|A\|_2 \|q_j\|_2 \]
\[ \|w_0\| \leq \|A\|_2 \leq \|A\|_F \]

From isometry of orthogonal projections,
\[ |h_{i,j}| \leq \|A\|_F \]

- \( h_{i,j} \) form Hessenberg Matrix
- Bound only computed once, valid for entire solve

**Evaluating the Impact of SDC in Numerical Methods**

J. Elliott, M. Hoemmen, F. Mueller, SC’13
Enabling Local Recovery from Local Faults

- Current recovery model: Local node failure, global kill/restart.
- Different approach:
  - App stores key recovery data in persistent local (per MPI rank) storage (e.g., buddy, NVRAM), and registers recovery function.
  - Upon rank failure:
    - MPI brings in reserve HW, assigns to failed rank, calls recovery fn.
    - App restores failed process state via its persistent data (& neighbors?).
    - All processes continue.
Motivation for LFLR:

- Current practice of Checkpoint/Restart is global response to single node (local) failure
  - Kill all processes (global terminate), then restart
  - Dependent on Global File system
  - SCR (LLNL) is fast, but adheres global recovery

- Single node failures are predominant
  - 85% on LLNL clusters (Moody et al. 2010)
  - 60-90% on Jaguar/Titan (ORNL)

- Need for scalable, portable and application agnostic solution
  - Local Failure Local Recovery Model (LFLR)
TRILINOS FRAMEWORK AND TOOLS
Trilinos Framework and Tools

- Changes since last year
  - User-focused web content moved from trilinos.sandia.gov to trilinos.org
    - Some links still need to be cleaned up
  - Moved to time-based, rather than feature-based releases (4 per year)
  - Updated tutorial delivery - Trilinos_tutorial
    - Student accounts accessible via ssh
    - Virtual machine image
    - Trilinos_tutorial is on Github
  - First collaborator agreements in place
    - A long way to go yet
Trilinos Framework and Tools

- Plans for Upcoming Year
  - Make public repo identical to developer repo
    - Put on Github, support pull request workflow
  - Move trilinos-user and trilinos-announce mail lists to trilinos.org
  - 12.0 release scheduled for April
    - Chance for packages to break backward compatibility
  - More collaboration agreements
  - Improve support for and integration of externally developed packages
TRILINOS SW ENGINEERING TECHNOLOGIES AND INTEGRATION
Trilinos Software Engineering Technologies and Integration

Progress in last year:
• TriBITS Hosted on Github
  • URL: https://github.com/TriBITSPub/TriBITS
  • Github issues, pull requests, etc.
  • Snapshotted into Trilinos (keep integrated).
    • http://trac.trilinos.org/wiki/TriBITSTrilinosDev
• TriBITS Documentation: Developers guide 170+ pages, build/test reference 30+ pages, overview document in progress …

Plans for Next Year:
• TriBITS System (IDEAS Project)
  • Partition TriBITS into lighter-weight framework(s), better support wrapping external software as a TriBITS package, etc.
  • Merge TriBITS concepts of Packages and TPLs => Construct larger meta-projects, build/install/test meta-projects in pieces, extract and build/install individual packages, (optionally) build Trilinos with TPLs, use export XXXConfig.cmake files as glue.
  • Standard installations of TriBITS
  • Overview and tutorials
  • Implementation of TriBITS Lifecycle Model in TriBITS system => Targeted metrics and testing of backward compatibility, valgrind, coverage, etc.
• TriBITS Lifecycle Model Adoption and Refinement: IDEAS, Trilinos, CASL
TRILINOS COMMUNITY 2.0
Trilinos Community 2.0

- GitHub, Atlassian:
  - Open source SW development, tools platforms.
  - Workflows for high-quality community SW product development.

- Trilinos value proposition:
  - Included these same things, but must re-evaluate.
  - Must address packages that want GitHub presence.
  - Must (IMO) move Trilinos itself to GitHub.

- New types of Trilinos packages (evolving):
  - Internal: Available only with Trilinos (traditional definition).
  - Exported: Developed in Trilinos repository, available externally.
  - Imported: Developed outside of Trilinos, available internally.
Trilinos Community 2.0

- Case studies:
  - TriBITS: Was an internal package, now external.
  - DTK: Has always been external.
  - KokkosCore: Is internal. Needs to be available externally.

- Issues to Resolve:
  - Package inclusion policies: Define for each package type.
  - Quality criteria: Contract between Trilinos and packages.
  - Workflows: Development, testing, documentation, etc.
  - Trilinos on GitHub: Evaluate.
  - Trilinos Value Proposition: Re-articulate Trilinos Strategic Goals implications.
WHY 2.0?
Day 1 of Package Life

- **CVS**: Each package is self-contained in Trilinos/package/ directory.
- **Bugzilla**: Each package has its own Bugzilla product.
- **Bonsai**: Each package is browsable via Bonsai interface.
- **Mailman**: Each Trilinos package, including Trilinos itself, has four mail lists:
  - package-checkins@software.sandia.gov
    - CVS commit emails. “Finger on the pulse” list.
  - package-developers@software.sandia.gov
    - Mailing list for developers.
  - package-users@software.sandia.gov
    - Issues for package users.
  - package-announce@software.sandia.gov
    - Releases and other announcements specific to the package.

- **New_package** (optional): Customizable boilerplate for
  - Autoconf/Automake/Doxygen/Python/Thyra/Epetra/TestHarness/Website
Day 1 of Package Life on GitHub & Atlassian

- **CVS -> Git**: Each package is self-contained, clonable.
- **Bonsai -> Browsing**: Git tools, local clone.
- **Mail lists -> Forums**: More diverse, interactive.
- **New_package ->** Stubbed out project.

- **Atlassian**: Even more advanced but costly.
  - Dev Tools: Git, mercurial, other tools.
  - Jira: Issue tracking.
  - Confluence: Collaboration, communication.
Software Platforms are Commodity

- Projects can start from Day 1 with high-quality environment, global visibility.
- Need to re-state the Trilinos value propositions from first principles.
Trilinos Community 2.0

- GitHub, Atlassian:
  - Open source SW development, tools platforms.
  - Workflows for high-quality community SW product development.

- Trilinos value proposition:
  - Included these same things, but must re-evaluate.
  - Must address packages that want GitHub presence.
  - Must (IMO) move Trilinos itself to GitHub.

- New types of Trilinos packages (evolving):
  - Internal: Available only with Trilinos (traditional definition).
  - Exported: Developed in Trilinos repository, available externally.
  - Imported: Developed outside of Trilinos, available internally.
Trilinos Community 2.0

- **Case studies:**
  - TriBITS: Was an internal package, now external.
  - DTK: Has always been external.
  - KokkosCore: Is internal. Needs to be available externally.
  - Tracked: STK, Percept, others?

- **Issues to Resolve:**
  - Package inclusion policies: Define for each package type.
  - Quality criteria: Contract between Trilinos and packages.
  - Workflows: Development, testing, documentation, etc.
  - Trilinos on GitHub: Evaluate.
  - Trilinos Value Proposition: Re-articulate Trilinos Strategic Goals implications.
MORE FROM 2006
Trilinos Interoperability Mechanisms (Acquired as Package Matures)

- Package builds under Trilinos configure scripts.
- Package accepts user data as Epetra or Thyra objects.
- Package accepts parameters from Teuchos ParameterLists.
- Package can be used via Thyra abstract solver classes.
- Package can use Epetra for private data.
- Package accesses solver services via Thyra interfaces.
- Package available via PyTrilinos, ForTrilinos, WebTrilinos.

Application can be built as part of a suite of packages; cross-package interfaces enable/disable automatically.

Applications using Epetra/Thyra can use package.

Applications using Teuchos ParameterLists can drive package.

Applications or other packages using Thyra can use package.

Package can then use other packages that implement Thyra interfaces.

Package can then use other packages that understand Epetra.

Package can be used with other Trilinos packages via Python, Fortran, Website.
## Sample Package Maturation Process

<table>
<thead>
<tr>
<th>Step</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package added to CVS: Import existing code or start with new_package.</td>
<td>ML CVS repository migrated into Trilinos (July 2002).</td>
</tr>
<tr>
<td>Mail lists, Bugzilla Product, Bonsai database created.</td>
<td>ml-announce, ml-users, ml-developers, ml-checkins, ml-regression @software.sandia.gov created, linked to CVS (July 2002).</td>
</tr>
<tr>
<td>Package builds with configure/make, Trilinos-compatible</td>
<td>ML adopts Autoconf, Automake starting from new_package (June 2003).</td>
</tr>
<tr>
<td>Epetra objects recognized by package.</td>
<td>ML accepts user data as Epetra matrices and vectors (October 2002).</td>
</tr>
<tr>
<td>Package accessible via Thyra interfaces.</td>
<td>ML adaptors written for TSFCore_LinOp (Thyra) interface (May 2003).</td>
</tr>
<tr>
<td>Package uses Epetra for internal data.</td>
<td>ML able to generate Epetra matrices. Allows use of AztecOO, Amesos, Ifpack, etc. as smoothers and coarse grid solvers (Feb-June 2004).</td>
</tr>
<tr>
<td>Package parameters settable via Teuchos ParameterList</td>
<td>ML gets manager class, driven via ParameterLists (June 2004).</td>
</tr>
</tbody>
</table>

### Startup Steps

### Maturation Steps
NewPackage Package

- NewPackage provides jump start to develop/integrate a new package.
- NewPackage is a “Hello World” program and website:
  - Simple but it does work with autotools.
  - Compiles and builds.
- NewPackage directory contains:
  - Commonly used directory structure: src, test, doc, example, config.
  - Working autotools files.
  - Documentation templates (doxygen).
  - Working regression test setup.
- Really cuts down on:
  - Time to integrate new package.
  - Variation in package integration details.
  - Development of website.
Internal vs. Exported vs. Imported

- **Internal:**
  - Interoperable with other packages, both up and downstream.
  - Clearly within the scope of Trilinos functionality domain, e.g., solvers of any kind.

- **Imported or exported?**
  - Snapshotting provides dual view of package, imported or exported.
  - Imported: Most commits come from developers outside of Trilinos.
  - Exported: “...” inside of Trilinos.
  - Imported: If you want to manage your own set of workflows or affiliate with another SW community.
  - Exported: If you don’t.

- **Other considerations?**
Common Look-and-feel Expectations

- Consistent data management practices.
- Consistent API styles.
- Testing and other quality metric thresholds, e.g., coverity.
- What else?
Trilinos Community 2.0 Next Steps

- Characterize, prioritize activities:
  - Package classifications.
  - Inclusion policies.
  - Workflows.
  - Quality criteria.

- Reformulate value propositions.
  - What is valuable about participating in Trilinos community.
    - What is already true.
    - What is partly true and needs to improve.
    - What is missing and should be added.

- Move Trilinos to GitHub.
  - Sanity test first. Move aggressively unless required.

- Revisit NewPackage concept.

- What else?
Summary

- Trilinos is an active project:
  - 10s of developers.
  - 100s of user-developers.
  - 1000s of users.

- HPC is in a highly disruptive phase:
  - Manycore/accelerator HW.
  - Increased opportunity for multiphysics/multiscale.
  - Resilience.
  - Community SW platforms: GitHub.com

- Trilinos R&D Response:
  - Programming environments.
  - Scalable, Resilient Algorithms, libraries.
  - New Trilinos Community Model and Package architecture.
  - New: Productivity (different talk)