Exceptional service in the national interest

Sandia National Laboratories

Getting Started with Trilinos

Michael A. Heroux
Sandia National Laboratories
USA

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Outline

Sandia National Laboratories

- Some (Very) Basics
- Overview of Trilinos: What it can do for you.
- Trilinos Software Organization.
- Overview of Packages.
- Package Management.
- Documentation, Building, Using Trilinos.

Online Resource

- Trilinos Website: https://trilinos.org
  - Portal to Trilinos resources.
  - Documentation.
  - Mail lists.
  - Downloads.
- WebTrilinos
  - Build & run Trilinos examples in your web browser!
  - Need username & password (will give these out later)
  - https://code.google.com/p/trilinos/wiki/TrilinosHandsOnTutorial
  - Example codes: https://code.google.com/p/trilinos/w/list

Sandia National Laboratories

WHY USE MATHEMATICAL LIBRARIES?

- A farmer had chickens and pigs. There was a total of 60 heads and 200 feet. How many chickens and how many pigs did the farmer have?
- Let x be the number of chickens, y be the number of pigs.
- Then:

$$x + y = 60$$
$$2x + 4y = 200$$

- From first equation $x = 60 - y$, so replace x in second equation:

$$2(60 - y) + 4y = 200$$

- Solve for y:

$$120 - 2y + 4y = 200$$
$$2y = 80$$
$$y = 40$$

- Solve for x: $x = 60 - 40 = 20.$
- The farmer has 20 chickens and 40 pigs.

- A restaurant owner purchased one box of frozen chicken and another box of frozen pork for $60. Later the owner purchased 2 boxes of chicken and 4 boxes of pork for $200. What is the cost of a box of frozen chicken and a box of frozen pork?
- Let x be the price of a box of chicken, y the price of a box of pork.
- Then:

$$x + y = 60$$
$$2x + 4y = 200$$

- From first equation $x = 60 - y$, so replace x in second equation:

$$2(60 - y) + 4y = 200$$

- Solve for y:

$$120 - 2y + 4y = 200$$
$$2y = 80$$
$$y = 40$$

- Solve for x: $x = 60 - 40 = 20$.
- A box of chicken costs $20 and a box of pork costs $40.

Sandia National Laboratories

Problem Statement

- A restaurant owner purchased one box of frozen chicken and another box of frozen pork for $60. Later the owner purchased 2 boxes of chicken and 4 boxes of pork for $200. What is the cost of a box of frozen chicken and a box of frozen pork?

Variables

- Let x be the price of a box of chicken, y the price of a box of pork.

Problem Setup

- Then:

$$x + y = 60$$
$$2x + 4y = 200$$

Solution Method

- From first equation $x = 60 - y$, so replace x in second equation:

$$2(60 - y) + 4y = 200$$

- Solve for y:

$$120 - 2y + 4y = 200$$
$$2y = 80$$
$$y = 40$$

- Solve for x: $x = 60 - 40 = 20$.

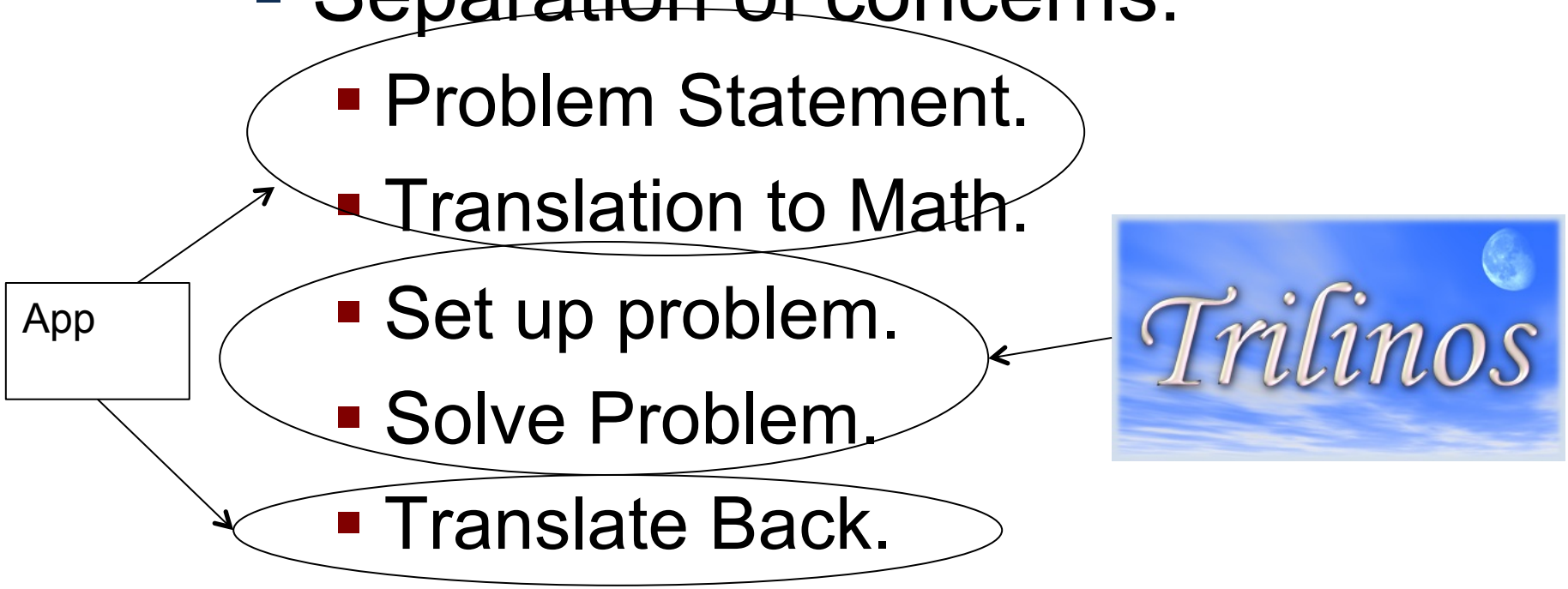
Translate Back

- A box of chicken costs $20. A box of pork costs $40.

Why Math Libraries?

Sandia National Laboratories

- Many types of problems.
- Similar Mathematics.
- Separation of concerns:
  - Problem Statement.
  - Translation to Math.
  - Set up problem.
  - Solve Problem.
  - Translate Back.



Importance of Math Libraries

- Computer solution of math problems is hard:
  - Floating point arithmetic not exact:
    - $1 + \varepsilon = 1$, for small $\varepsilon > 0$.
    - $(a + b) + c$ not always equal to $a + (b + c)$.
  - High fidelity leads to large problems: 1M to 10B equations.
  - Clusters require coordinated solution across 100 – 1M processors.
- Sophisticated solution algorithms and libraries leveraged:
  - Solver expertise highly specialized, expensive.
  - Write code once, use in many settings.
- Trilinos is a large collection of state-of-the-art work:
  - The latest in scientific algorithms.
  - Leading edge software design and architecture.

What is Trilinos?

Sandia National Laboratories

- Object-oriented software framework for…
- Solving big complex science & engineering problems
- More like LEGO™ bricks than Matlab™

Sandia National Laboratories

BACKGROUND/MOTIVATION

Trilinos

R&D 100

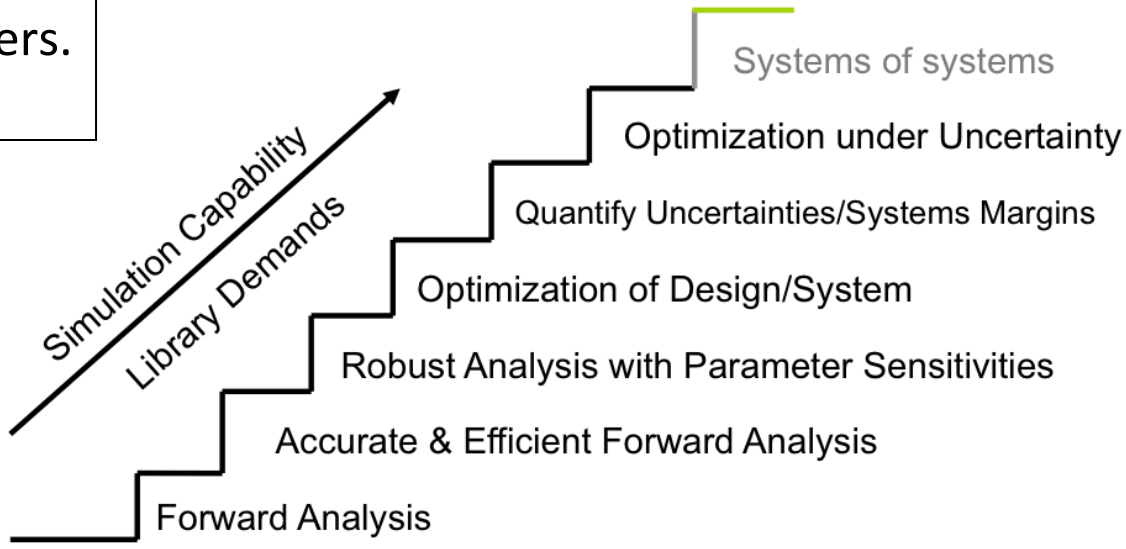
Laptops to
Leadership systems

Optimal Kernels to Optimal Solutions:

- Geometry, Meshing
- Discretizations, Load Balancing.
- Scalable Linear, Nonlinear, Eigen, Transient, Optimization, UQ solvers.
- Scalable I/O, GPU, Manycore

- 60 Packages.
- Other distributions:
  - Cray LIBSCI
  - Public repo.
- Thousands of Users.
- Worldwide distribuion.

Transforming Computational Analysis To Support High Consequence Decisions



Each stage requires *greater performance* and *error control* of prior stages:
Always will need: more accurate and scalable methods.
more sophisticated tools.

Trilinos Strategic Goals

Sandia National Laboratories

- Scalable Computations: As problem size and processor counts increase, the cost of the computation will remain nearly fixed.
- Hardened Computations: Never fail unless problem essentially intractable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.
- Full Vertical Coverage: Provide leading edge enabling technologies through the entire technical application software stack: from problem construction, solution, analysis and optimization.

Algorithmic Goals

- *Grand* Universal Interoperability: All Trilinos packages, and important external packages, will be interoperable, so that any combination of packages and external software (e.g., PETSc, Hypre) that makes sense algorithmically will be possible within Trilinos.
- Universal Accessibility: All Trilinos capabilities will be available to users of major computing environments: C++, Fortran, Python and the Web, and from the desktop to the latest scalable systems.
- Universal Solver RAS: Trilinos will be:
  - Reliable: Leading edge hardened, scalable solutions for each of these applications
  - Available: Integrated into every major application at Sandia
  - Serviceable: "Self-sustaining".

Software Goals

Capability Leaders: Layer of Proactive Leadership

- Areas:
  - Framework, Tools & Interfaces (J. Willenbring).
  - Software Engineering Technologies and Integration (R. Bartlett).
  - Discretizations (P. Bochev).
  - Geometry, Meshing & Load Balancing (K. Devine).
  - Parallel Programming Models (H.C. Edwards)
  - Linear Algebra Services (M. Hoemmen).
  - Linear & Eigen Solvers (J. Hu).
  - Nonlinear, Transient & Optimization Solvers (A. Salinger).
  - Scalable I/O: (R. Oldfield)
  - User Experience: (W. Spotz)
- Each leader provides strategic direction across all Trilinos packages within area.

Unique features of Trilinos

Sandia National Laboratories

- Huge library of algorithms
  - Linear and nonlinear solvers, preconditioners, …
  - Optimization, transients, sensitivities, uncertainty, …
- Growing support for multicore & hybrid CPU/GPU
  - Built into the new Tpetra linear algebra objects
    - Therefore into iterative solvers with zero effort!
  - Unified intranode programming model
    - Spreading into the whole stack:
      - Multigrid, sparse factorizations, element assembly…
  - X: Threads (CPU, Intel Xeon Phi, CUDA on GPU)
- Support for mixed and arbitrary precisions
  - Don’t have to rebuild Trilinos to use it
- Support for huge (> 2B unknowns) problems

Applications

Sandia National Laboratories

- All kinds of physical simulations:
  - Structural mechanics (statics and dynamics)
  - Circuit simulations (physical models)
  - Electromagnetics, plasmas, and superconductors
  - Combustion and fluid flow (at macro- and nanoscales)
- Coupled multiphysics, multiscale models
- Data and graph analysis

Trilinos Current Release

Sandia National Laboratories

- Trilinos 11.12 current.
  - 55 packages.
  - One new package: Muelu (next-gen multigrid).
- Website: trilinos.org.

- Trilinos 11.14 almost out the door.
- We will use it today.

Sandia National Laboratories

Trilinos software organization

Trilinos is composed of packages

Sandia National Laboratories

- Not a monolithic piece of software
- Each package
  - Has its own development team and management
  - Makes its own decisions about algorithms, coding style, etc.
  - May or may not depend on other Trilinos packages
  - May even have a different license (most are 3-term BSD) or release status
  - Benefits from Trilinos build and test infrastructure
- Trilinos is not “indivisible”
  - You don’t need all of Trilinos to get things done
  - Don’t feel overwhelmed by large number (~60) of packages!
  - Any subset of packages can be combined and distributed
- Trilinos top layer framework (TriBITS)
  - Manages package dependencies
  - Runs packages’ tests nightly, and on every check-in
  - Useful: spun off from Trilinos into a separate project

Why packages?

Sandia National Laboratories

- Users decide how much of Trilinos they want to use
  - Only use and build the packages you need
  - Mix and match Trilinos components with your own, e.g.,
    - Trilinos sparse matrices with your own linear solvers
    - Your sparse matrices with Trilinos' linear solvers
    - Trilinos sparse matrices & linear solvers with your nonlinear solvers
- Popular packages (e.g., ML, Zoltan) keep their "brand"
  - But benefit from Trilinos build & test infrastructure
- Reflects organization of research / development teams
  - Easy to turn a research code into a new package
  - Small teams with minimal interference between teams
- TriBITS build system supports external packages!
  - Data Transfer Kit: https://github.com/CNERG/DataTransferKit
  - Need not live in Trilinos' repository or have Trilinos' license

Solver Software Stack

Phase I packages: SPMD, int/double

Phase II packages: Templated

Optimization Unconstrained: Constrained:	Find $u \in \Re^n$ that minimizes $g(u)$ Find $x \in \Re^m$ and $u \in \Re^n$ that minimizes $g(x,u)$ s.t. $f(x,u) = 0$	MOOCHO
Bifurcation Analysis	Given nonlinear operator $F(x,u) \in \Re^{n+m}$ For $F(x,u) = 0$ find space $u \in U \ni \frac{\partial F}{\partial x}$	LOCA
Transient Problems DAEs/ODEs:	Solve $f(\dot{x}(t), x(t), t) = 0$ $t \in [0,T], x(0) = x_0, \dot{x}(0) = x_0'$ for $x(t) \in \Re^n, t \in [0,T]$	Rythmos
Nonlinear Problems	Given nonlinear operator $F(x) \in \Re^m \rightarrow \Re$ Solve $F(x) = 0$ $x \in \Re^n$	NOX
Linear Problems Linear Equations: Eigen Problems:	Given Linear Ops (Matrices) $A, B \in \Re^{m \times n}$ Solve $Ax = b$ for $x \in \Re^n$ Solve $A\nu = \lambda B\nu$ for (all) $\nu \in \Re^n$, $\lambda \in$	Anasazi Ifpack, ML, etc... AztecOO
Distributed Linear Algebra Matrix/Graph Equations: Vector Problems:	Compute $y = Ax; A = A(G); A \in \Re^{m \times n}, G \in \Im^{m \times n}$ Compute $y = \alpha x + \beta w; \alpha = \langle x, y \rangle; x, y \in \Re^n$	Epetra Teuchos

Sensitivities (Automatic Differentiation: Sacado)

Solver Software Stack

Phase I packages | Phase II packages | Phase III packages: Manycore*, templated

Optimization Unconstrained: Constrained:	Find $u \in \Re^n$ that minimizes $g(u)$ Find $x \in \Re^m$ and $u \in \Re^n$ that minimizes $g(x,u)$ s.t. $f(x,u) = 0$	MOOCHO
Bifurcation Analysis	Given nonlinear operator $F(x,u) \in \Re^{n+m}$ For $F(x,u) = 0$ find space $u \in U \ni \frac{\partial F}{\partial x}$	LOCA | T-LOCA
Transient Problems DAEs/ODEs:	Solve $f(\dot{x}(t), x(t), t) = 0$ $t \in [0,T], x(0) = x_0, \dot{x}(0) = x_0'$ for $x(t) \in \Re^n, t \in [0,T]$	Rythmos
Nonlinear Problems	Given nonlinear operator $F(x) \in \Re^m \rightarrow \Re$ Solve $F(x) = 0$ $x \in \Re^n$	NOX | T-NOX
Linear Problems Linear Equations: Eigen Problems:	Given Linear Ops (Matrices) $A, B \in \Re^{m \times n}$ Solve $Ax = b$ for $x \in \Re^n$ Solve $A\nu = \lambda B\nu$ for (all) $\nu \in \Re^n$, $\lambda \in$	Anasazi AztecOO Ifpack, ML, etc... | Belos* Ifpack2*, Muelu*,etc.
Distributed Linear Algebra Matrix/Graph Equations: Vector Problems:	Compute $y = Ax; A = A(G); A \in \Re^{m \times n}, G \in \Im^{m \times n}$ Compute $y = \alpha x + \beta w; \alpha = \langle x, y \rangle; x, y \in \Re^n$	Epetra | Tpetra* Kokkos* Teuchos

Sensitivities (Automatic Differentiation: Sacado)

Trilinos Package Summary

Sandia National Laboratories

	Objective	Package(s)
Discretizations	Meshing & Discretizations	STK, Intrepid, Pamgen, Sundance, ITAPS, Mesquite
	Time Integration	Rythmos
Methods	Automatic Differentiation	Sacado
	Mortar Methods	Moertel
Services	Linear algebra objects	Epetra, Tpetra, Kokkos, Xpetra
	Interfaces	Thyra, Stratimikos, RTOp, FEI, Shards
	Load Balancing	Zoltan, Isorropia, Zoltan2
	“Skins”	PyTrilinos, WebTrilinos, ForTrilinos, Ctrilinos, Optika
	C++ utilities, I/O, thread API	Teuchos, EpetraExt, Kokkos, Triutils, ThreadPool, Phalanx, Trios
Solvers	Iterative linear solvers	AztecOO, Belos, Komplex
	Direct sparse linear solvers	Amesos, Amesos2, ShyLU
	Direct dense linear solvers	Epetra, Teuchos, Pliris
	Iterative eigenvalue solvers	Anasazi, Rbgen
	ILU-type preconditioners	AztecOO, IFPACK, Ifpack2, ShyLU
	Multilevel preconditioners	ML, CLAPS, **Muelu**
	Block preconditioners	Meros, Teko
	Nonlinear system solvers	NOX, LOCA, Piro
	Optimization (SAND)	MOOCHO, Aristos, TriKota, Globipack, Optipack
	Stochastic PDEs	Stokhos

Interoperability vs. Dependence

("Can Use") ("Depends On")

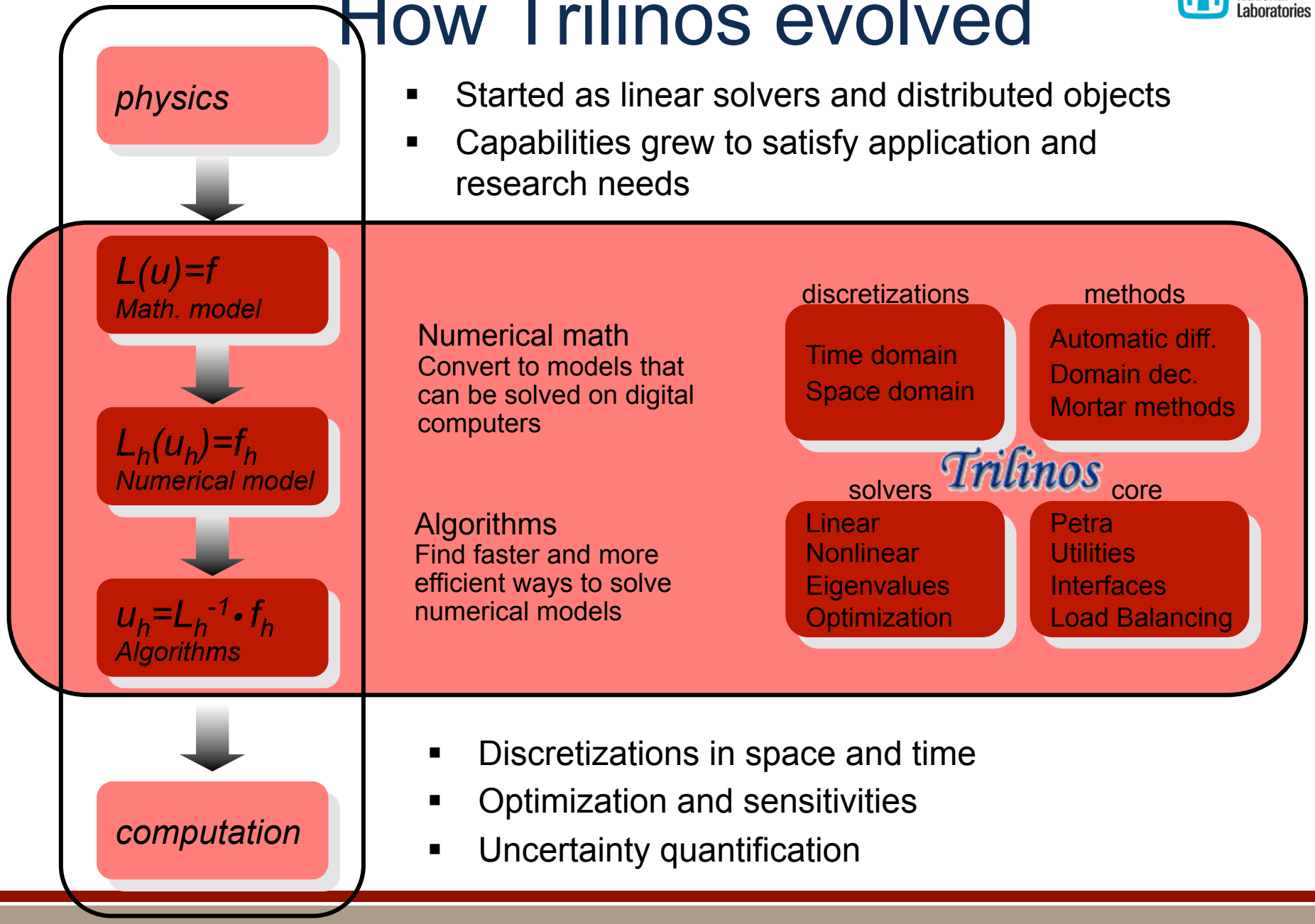
- Although most Trilinos packages have no explicit dependence, often packages must interact with *some* other packages:
- NOX needs operator, vector and linear solver objects.
- Belos needs preconditioner, matrix, operator and vector objects.
- Interoperability is enabled at configure time.

- Trilinos `cmake` system is vehicle for:
- Establishing interoperability of Trilinos components…
- Without compromising individual package autonomy.
- Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES option
- Architecture supports simultaneous development on many fronts.

Sandia National Laboratories

How Trilinos evolved

- Started as linear solvers and distributed objects
- Capabilities grew to satisfy application and research needs



- Discretizations in space and time
- Optimization and sensitivities
- Uncertainty quantification

Sandia National Laboratories

Whirlwind Tour of Packages

Sandia National Laboratories

Whirlwind Tour of Packages

Discretizations Methods Core Solvers

Teuchos

Sandia National Laboratories

- Portable utility package of commonly useful tools

- ParameterList: nested (key, value) database (more later)
- Generic LAPACK and BLAS wrappers
- Local dense matrix and vector classes
- Memory management classes (more later)
- Scalable parallel timers and statistics
- Support for generic algorithms (traits classes)
- Help make Trilinos work on as many platforms as possible
- Protect algorithm developers from platform differences
- Not all compilers could build Boost in the mid-2000s
- BLAS and LAPACK Fortran vs. C calling conventions
- Different sizes of integers on different platforms
- You'll see this package a lot

Package lead: Roscoe Barlett (many developers)

Trilinos Common Language: Petra

- "Common language" for distributed linear algebra objects (operator, sparse matrix, dense vectors)

- Petra[1] provides distributed matrix and vector services
- Object model
- Describes basic user and support classes in UML, independent of language/implementation
- Describes objects and relationships to build and use matrices, vectors and graphs
- Has 3 implementations under active development
  - Epetra, Tpetra, Xpetra

[1]Petra is Greek for "foundation".

Petra Implementations

Sandia National Laboratories

- Epetra (Essential Petra):
  - Earliest and most heavily used
  - C++ circa 1998 (“C+/- compilers” OK)
  - Real, double-precision arithmetic
  - Interfaces accessible to C and Fortran users

- Tpetra (Templated Petra):
  - C++ circa mid-2000s (no C++11)
  - Supports arbitrary scalar and index types via templates
    - Arbitrary- and mixed-precision arithmetic
    - 64-bit indices for solving problems with >2 billion unknowns
  - Hybrid MPI / shared-memory parallel
    - Supports multicore CPU and hybrid CPU/GPU
    - Built on Kokkos shared-memory parallel programming model

- Xpetra (Templated Petra): Developer API to both Epetra/Tpetra.

Package leads: Mike Heroux, Mark Hoemmen, Andrey Prokopenko (many developers)

Two solver stacks: Epetra & Tpetra

Sandia National Laboratories

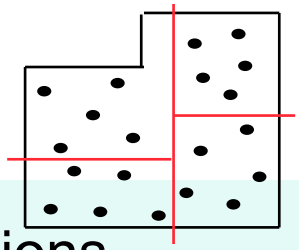
- Many packages built on Epetra linear algebra interface
  - Common “solver stack” for Epetra sparse matrices and vectors
- Users want features that break interfaces
  - Support for solving huge problems (> 2B entities)
  - Arbitrary and mixed precision
  - Hybrid (MPI+X) parallelism (← most radical interface changes)
- Trilinos’ customers value backwards compatibility
- We decided to build a new stack using Tpetra
  - Work on interfaces to support MPI+X is going into Tpetra stack
  - Epetra has gotten some support for huge problems (“Epetra64”)
  - Some packages can work with either Epetra or Tpetra
    - Iterative linear solvers & eigensolvers (Belos, Anasazi)
    - Multilevel preconditioning (MueLu), sparse direct (Amesos2)

Kokkos: Shared-memory parallelism

- Manycore parallel programming model & data structures
- Kokkos parallel dispatch and data structures
  - Discussed in depth this afternoon.
  - Multi-dimensional arrays, hash table, sparse graph & matrix
  - Hide physical data layout & target it to the hardware
- "Pretty Good Kernels"
  - Computational kernels for sparse & dense matrices
  - Written generically to Kokkos, not to specific hardware
    - "How I learned to stop worrying and love CSR"
  - Replaceable with vendor-optimized libraries

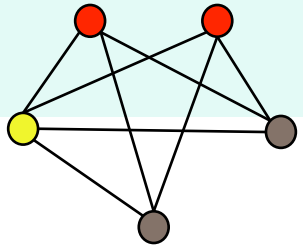
Developer: Carter Edwards, Mark Hoemmen, Dan Sunderland, Christian Trott

Zoltan(2)

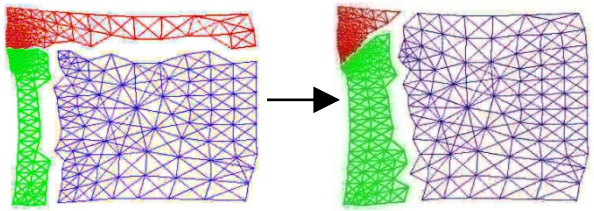


Sandia National Laboratories

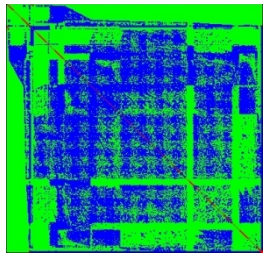
- Data Services for Dynamic Applications

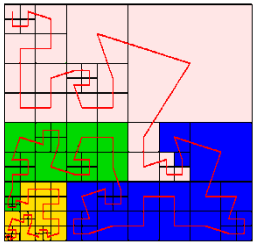


- Dynamic load balancing
- Graph coloring
- Data migration
- Matrix ordering

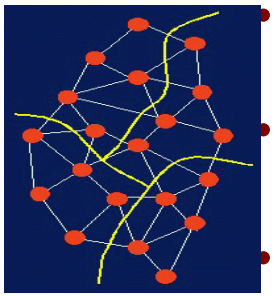


- Partitioners:

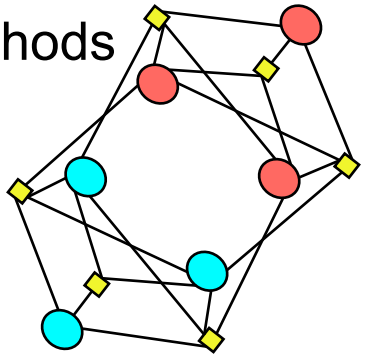




Geometric (coordinate-based) methods:
Recursive Coordinate Bisection
Recursive Inertial Bisection
Space Filling Curves
Refinement-tree Partitioning



- Hypergraph and graph (connectivity-based) methods



- Isorropia package: interface to Epetra objects
- Zoltan2: interface to Tpetra objects

Developers: Karen Devine, Erik Boman, Siva Rajamanickam, Michael Wolf

Thyra

Sandia National Laboratories

- Abstract linear algebra interfaces
- Offers flexibility through abstractions to algorithm developers
- Linear solvers (Direct, Iterative, Preconditioners)
  - Use abstraction of basic matrix & vector operations
  - Can use any concrete linear algebra library (Epetra, Tpetra, …)
- Nonlinear solvers (Newton, etc.)
  - Use abstraction of linear solve
  - Can use any concrete linear solver library and preconditioners
- Transient/DAE solvers (implicit)
  - Use abstraction of nonlinear solve
- Thyra is how Stratimikos talks to data structures & solvers

Developers: Roscoe Bartlett, Kevin Long

“Skins”

Sandia National Laboratories

- PyTrilinos provides Python access to Trilinos packages
- Uses SWIG to generate bindings.
- Support for many packages

Developer: Bill Spotz

- CTrilinos: C wrapper (mostly to support ForTrilinos).
- ForTrilinos: OO Fortran interfaces.

Developers: Nicole Lemaster, Damian Rouson

- WebTrilinos: Web interface to Trilinos
- Generate test problems or read from file.
- Generate C++ or Python code fragments and click-run.
- Hand modify code fragments and re-run.
- Will use during hands-on.

Developers: Ray Tuminaro, Jonathan Hu, Marzio Sala, Jim Willenbring

Anasazi

- Next-generation iterative eigensolvers

- Decouples algorithms from linear algebra objects
  - Like Belos, except that Anasazi came first
- Block eigensolvers for accurate cluster resolution
  - This motivated Belos' block & "pseudoblock" linear solvers
- Can solve
  - Standard ($AX = \Lambda X$) or generalized ($AX = BX\Lambda$)
  - Hermitian or not, real or complex
- Algorithms available
  - Block Krylov-Schur (most like ARPACK's IR Arnoldi)
  - Block Davidson (improvements in progress)
  - Locally Optimal Block-Preconditioned CG (LOBPCG)
  - Implicit Riemannian Trust Region solvers
  - Scalable orthogonalizations (e.g., TSQR, SVQB)

Developers: Heidi Thornquist, Mike Heroux, Chris Baker, Rich Lehoucq, Ulrich Hetmaniuk, Mark Hoemmen

NOX: Nonlinear Solvers

Sandia National Laboratories

- Suite of nonlinear solution methods

Broyden's Method

$$M_B = f(x_c) + B_c d$$

Newton's Method

$$M_N = f(x_c) + J_c d$$

Tensor Method

$$M_T = f(x_c) + J_c d + \frac{1}{2} T_c dd$$

Jacobian Estimation
- Graph Coloring
- Finite Difference
- Jacobian-Free Newton-Krylov

Globalizations

Line Search
Interval Halving
Quadratic
Cubic
More'-Thuente

Trust Region
Dogleg
Inexact Dogleg

Implementation
- Parallel
- Object-oriented C++
- Independent of the linear algebra implementation!

Developers: Tammy Kolda, Roger Pawlowski

LOCA: Continuation problems

Sandia National Laboratories

- Solve parameterized nonlinear systems $F(x, \lambda) = 0$
- Identify "interesting" nonlinear behavior
  - Turning points
    - Buckling of a shallow arch under symmetric load
    - Breaking away of a drop from a tube
    - Bursting of a balloon at a critical volume
  - Bifurcations, e.g., Hopf or pitchfork
    - Vortex shedding (e.g., turbulence near mountains)
    - Flutter in airplane wings; electrical circuit oscillations
- LOCA uses Trilinos components
  - NOX to solve nonlinear systems
  - Anasazi to solve eigenvalue problems
  - AztecOO or Belos to solve linear systems

Developers: Andy Salinger, Eric Phipps

MOOCHO & Aristos

Sandia National Laboratories

- MOOCHO: Multifunctional Object-Oriented arCHitecture for Optimization
  - Large-scale invasive simultaneous analysis and design (SAND) using reduced space SQP methods.

Developer: Roscoe Bartlett

- Aristos: Optimization of large-scale design spaces
  - Invasive optimization approach
  - Based on full-space SQP methods
  - Efficiently manages inexactness in the inner linear solves
- ***New:*** ROL under development, available soon.

Developer: Denis Ridzal

Whirlwind Tour of Packages

Core Utilities
Discretizations Methods Solvers

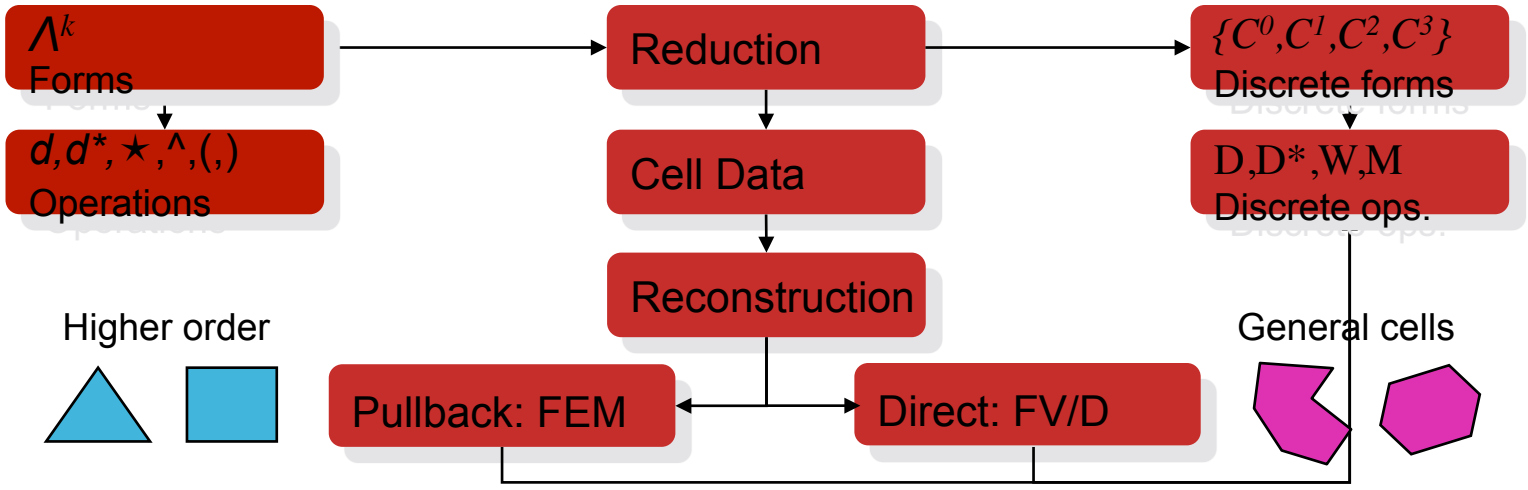
Intrepid

Interoperable Tools for Rapid Development of Compatible Discretizations

Sandia National Laboratories

Intrepid offers an innovative software design for compatible discretizations:

- Access to finite {element, volume, difference} methods using a common API
- Supports hybrid discretizations (FEM, FV and FD) on unstructured grids
- Supports a variety of cell shapes:
  - Standard shapes (e.g., tets, hexes): high-order finite element methods
  - Arbitrary (polyhedral) shapes: low-order mimetic finite difference methods
- Enables optimization, error estimation, V&V, and UQ using fast invasive techniques (direct support for cell-based derivative computations or via automatic differentiation)



Developers: Pavel Bochev and Denis Ridzal

Rythmos

Sandia National Laboratories

- Numerical time integration methods
- "Time integration" == "ODE solver"
- Supported methods include
  - Backward & Forward Euler
  - Explicit Runge-Kutta
  - Implicit BDF
- Operator splitting methods & sensitivities

Developers: Glen Hansen, Roscoe Bartlett, Todd Coffey

Sierra ToolKit (STK) modules overview

Sandia National Laboratories

Parallel-consistent Mesh database

- Heterogeneous element types
- Unstructured

Algorithm-Support (AlgSup)

- Multi-threaded execution of bucket-loop algorithms

Search

- Proximity, mesh independent

Linsys, IO, Rebalance

- Bridges from mesh-data to external capabilities
- Built optionally

Util

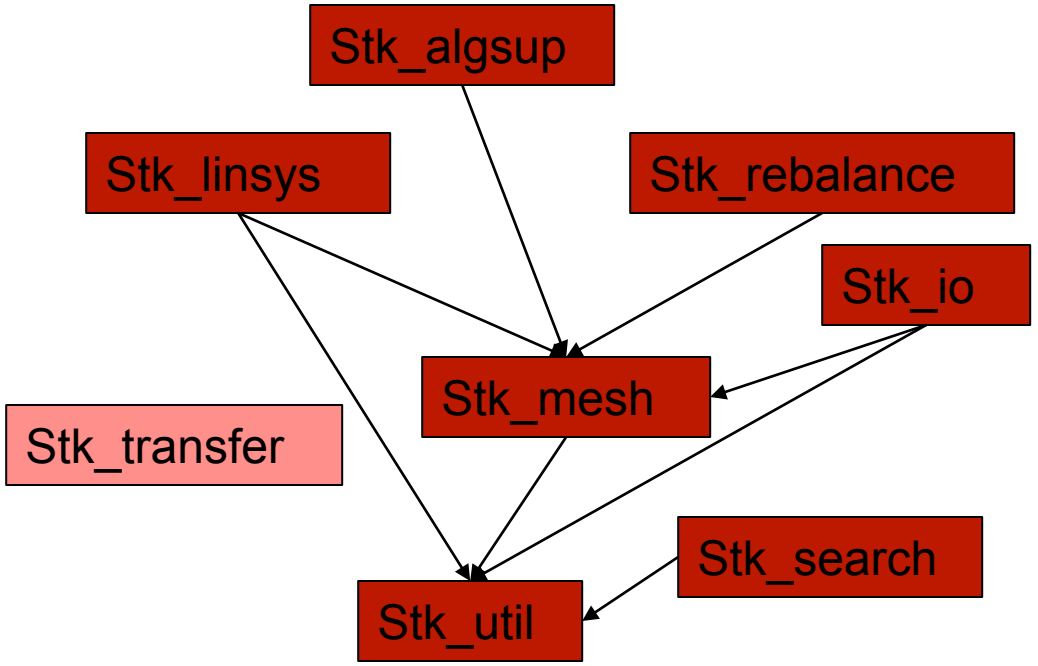
- Everything depends on util directly or indirectly

Transfer

- Not yet implemented

Dependency diagram:

- Arrows point towards a module that is used (depended on) by another module.



Sandia National Laboratories

Whirlwind Tour of Packages

Discretizations Methods Core Solvers

Sacado: Automatic Differentiation

- Automatic differentiation tools optimized for element-level computation

- Applications of AD: Jacobians, sensitivity and uncertainty analysis, …
- Uses C++ templates to compute derivatives
  - You maintain one templated code base; derivatives don't appear explicitly
- Provides three forms of AD
  - Forward Mode: $(x,\ V) \longrightarrow \left(f,\ \frac{\partial f}{\partial x} V\right)$
    - Propagate derivatives of intermediate variables w.r.t. independent variables forward
    - Directional derivatives, tangent vectors, square Jacobians, ∂f / ∂x when **m ≥ n**
  - Reverse Mode: $(x,\ W) \longrightarrow \left(f,\ W^T \frac{\partial f}{\partial x}\right)$
    - Propagate derivatives of dependent variables w.r.t. intermediate variables backwards
    - Gradients, Jacobian-transpose products (adjoints), ∂f / ∂x when **n > m**.
  - Taylor polynomial mode: $x(t) = \sum_{k=0}^{d} x_k t^k \longrightarrow \sum_{k=0}^{d} f_k t^k = f(x(t)) + O(t^{d+1}),\quad f_k = \frac{1}{k!}\frac{d^k}{dt^k} f(x(t))$
  - Basic modes combined for higher derivatives

Developer: Eric Phipps

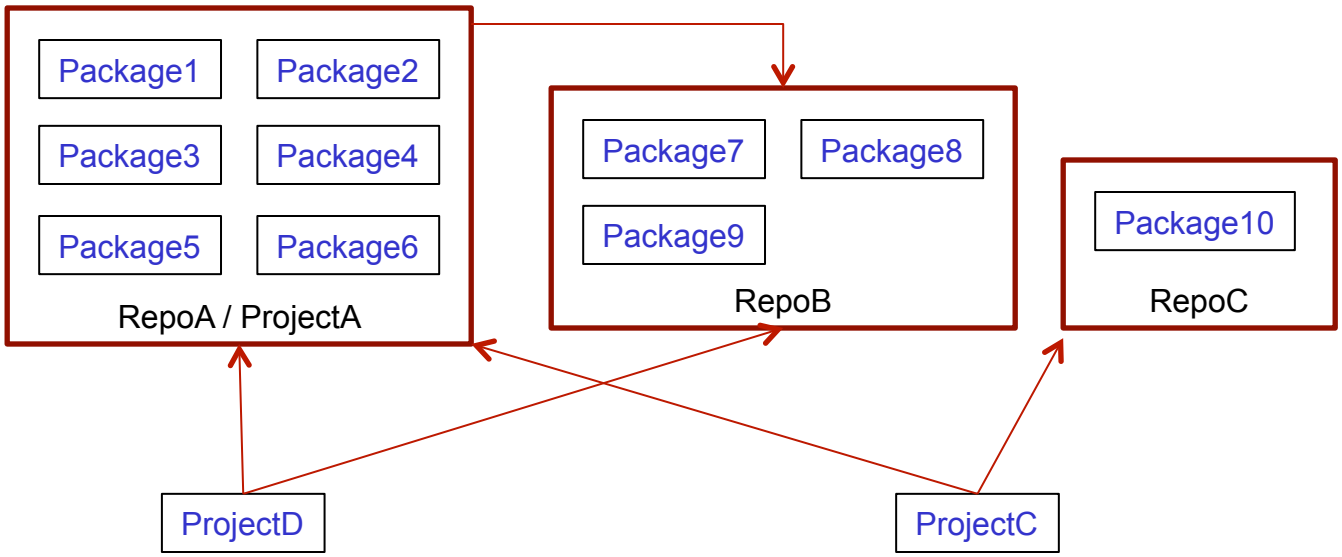
Trilinos' package management

TriBITS: Trilinos/Tribal Build, Integrate, Test System

Sandia National Laboratories

- Based on CMake, CTest, & CDash (Kitware open-source toolset)
  - Developed during Trilinos' move to CMake
  - Later extended for use in CASL projects (e.g., VERA) & SCALE
- Partitions a project into packages
  - Common CMake build & test infrastructure across packages
  - Handles dependencies between packages
- Integrated support for MPI, CUDA, & third-party libraries (TPLs)
- Multi-repository development
  - Can depend on packages in external repositories
  - Handy for mixing open-source & closed-source packages
- Test driver (tied into CDash)
  - Runs nightly & continuous integration
  - Helps target which package caused errors
- Pre-push synchronous continuous integration testing
  - Developers must use Python checkin-test.py script to push
  - The script enables dependent packages, & builds & runs tests
  - Also automates asynchronous continuous integration tests
- Lots of other features!

TriBITS: Meta Project, Repository, Packages



Current TriBITS features:

- Flexibly aggregate packages from different repositories into big projects
- Can use TriBITS in stand-alone projects (independent of Trilinos)
- In use by CASL VERA software, and several other CASL-related software packages
- Tool to manage multiple git repositories

Future changes/additions to TriBITS

- Combine concepts of TPLs & packages to allow flexible configuration & building
- TribitsExampleProject
- Trilinos-independent TriBITS documentation
- Provide open access to TribitsExampleProject and therefore TriBITS

Sandia National Laboratories

Building your application with Trilinos

Building your application with Trilinos

Sandia National Laboratories

If you are using Makefiles:

- Makefile.export system

If you are using CMake:

- CMake FIND_PACKAGE

Trilinos helps you link it with your application

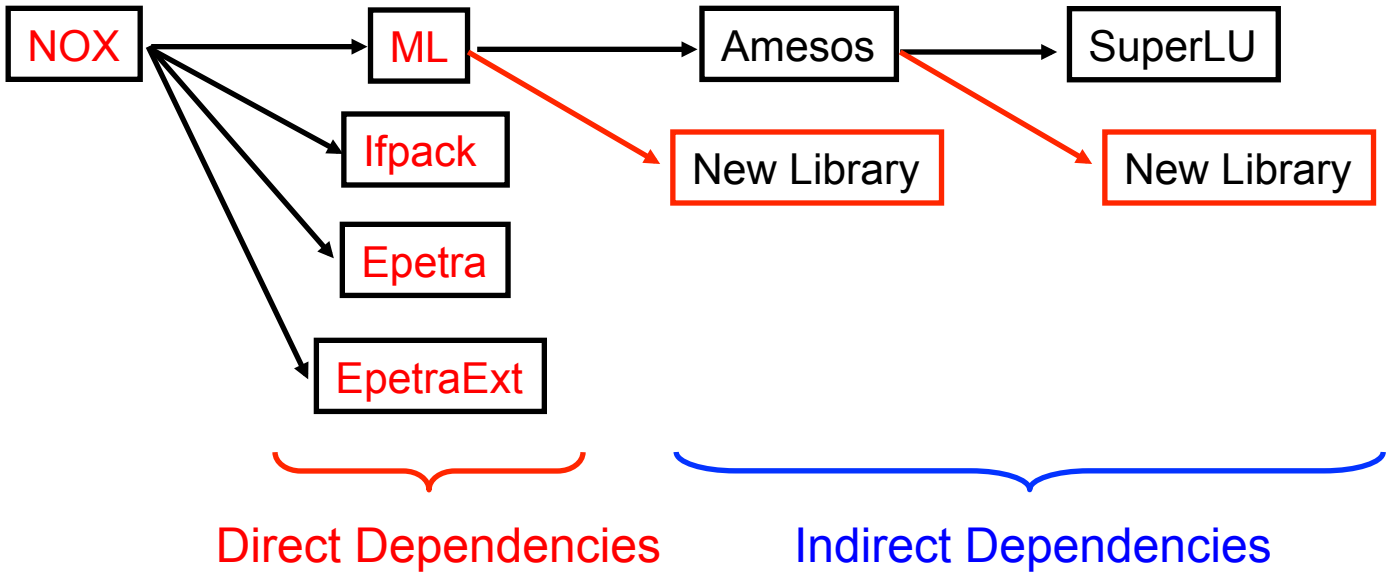
Sandia National Laboratories

- Library link order
  - -lnoxepetra -lnox –lepetra –lteuchos –lblas –llapack
  - Order matters!
  - Optional package dependencies affect required libraries
- Using the same compilers that Trilinos used
  - g++ or icc or icpc or …?
  - mpiCC or mpCC or mpicxx or … ?
- Using the same libraries that Trilinos used
  - Using Intel's MKL requires a web tool to get the link line right
  - Trilinos remembers this so you don't have to
- Consistent build options and package defines:
  - g++ -g –O3 –D HAVE_MPI –D _STL_CHECKED

Why is this hard? Why doesn't "-ltrilinos" work?

Sandia National Laboratories

- Trilinos has LOTS of packages
- Top-level packages might get new package dependencies indirectly, without knowing it:



Better to let Trilinos tell you what libraries you need!
It already does the work for you.

Using CMake to build with Trilinos

Sandia National Laboratories

- CMake: Cross-platform build system
  - Similar function as the GNU Autotools
- Building Trilinos requires CMake
- You don't have to use CMake to use Trilinos
- But if you do: FIND_PACKAGE(Trilinos …)
  - Example: https://code.google.com/p/trilinos/wiki/CMakeFindPackageTrilinosExample
- I find this much easier than writing Makefiles

Using the Makefile.export system

Sandia National Laboratories

```
#
# A Makefile that your application can use if you want to build with Epetra.
#
# You must first set the TRILINOS_INSTALL_DIR variable.

# Include the Trilinos export Makefile for the Epetra package.
include $(TRILINOS_INSTALL_DIR)/include/Makefile.export.Epetra

# Add the Trilinos installation directory to the library and header search paths.
LIB_PATH = $(TRILINOS_INSTALL_DIR)/lib
INCLUDE_PATH = $(TRILINOS_INSTALL_DIR)/include $(CLIENT_EXTRA_INCLUDES)

# Set the C++ compiler and flags to those specified in the export Makefile.
# This ensures your application is built with the same compiler and flags
# with which Trilinos was built.
CXX = $(EPETRA_CXX_COMPILER)
CXXFLAGS = $(EPETRA_CXX_FLAGS)

# Add the Trilinos libraries, search path, and rpath to the
# linker command line arguments
LIBS = $(CLIENT_EXTRA_LIBS) $(SHARED_LIB_RPATH_COMMAND) \
  $(EPETRA_LIBRARIES) \
  $(EPETRA_TPL_LIBRARIES) \
  $(EPETRA_EXTRA_LD_FLAGS)

#
# Rules for building executables and objects.
#
%.exe : %.o $(EXTRA_OBJS)
        $(CXX) -o $@ $(LDFLAGS) $(CXXFLAGS) $< $(EXTRA_OBJS) -L$(LIB_PATH) $(LIBS)

%.o : %.cpp
        $(CXX) -c -o $@ $(CXXFLAGS) -I$(INCLUDE_PATH) $(EPETRA_TPL_INCLUDES) $<
```

Sandia National Laboratories

Documentation, downloading, & building Trilinos

How do I learn more?

- Documentation:
  - Per-package documentation: http://trilinos.org/packages/
  - Other material on Trilinos website: http://trilinos.org/
  - Trilinos Wiki with many runnable examples: https://code.google.com/p/trilinos/wiki/
- E-mail lists: http://trilinos.org/mail_lists.html
- Annual user meetings and other tutorials:
  - Trilinos User Group (TUG) meeting and tutorial
    - Late October, or early November at SNL / NM
    - Talks available for download (slides and video):
      - http://trilinos.sandia.gov/events/trilinos_user_group_201<N>
      - Where N is 0, 1, 2, 3
  - European TUG meetings (once yearly)
    - This one: Paris-Saclay.
    - Next one: Looking for host.

How do I get Trilinos?

- Current release (11.12) available for download
  - http://trilinos.sandia.gov/download/trilinos-11.12.html
  - Source tarball with sample build scripts
- Public (read-only) git repository
  - http://trilinos.org/publicRepo/
  - Development version, updated ~ nightly
- Cray packages recent releases of Trilinos
  - http://www.nersc.gov/users/software/programming-libraries/math-libraries/trilinos/
  - `$ module load trilinos`
  - Recommended for best performance on Cray machines
- Most packages under BSD license
  - A few packages are LGPL

How do I build Trilinos?

Sandia National Laboratories

- Need C and C++ compiler and the following tools:
  - CMake (version >= 2.8), BLAS, & LAPACK
- Optional software:
  - MPI (for distributed-memory parallel computation)
  - Many other third-party libraries
- You may need to write a short configure script
  - Sample configure scripts in sampleScripts/
  - Find one closest to your software setup, & tweak it
- Build sequence looks like GNU Autotools
  1. Invoke your configure script, that invokes CMake
  2. `make -j<N> && make -j<N> install`
- Documentation:
  - TrilinosBuildQuickRef.* in Trilinos source directory
  - http://trilinos.sandia.gov/Trilinos11CMakeQuickstart.txt
  - https://code.google.com/p/trilinos/wiki/BuildScript

Sandia National Laboratories

Questions?

Hands-on tutorial

- WebTrilinos
  - Build & run Trilinos examples in your web browser!
  - Need username & password (will give these out later)
  - https://code.google.com/p/trilinos/wiki/TrilinosHandsOnTutorial
  - Example codes: https://code.google.com/p/trilinos/w/list

Other options to use Trilinos

- Virtual machine
  - Install VirtualBox, download VM file, and run it
  - Same environment as student shell accounts
  - We won't cover this today, but feel free to try it
- Build Trilinos yourself on your computer
  - Prerequisites:
    - C++ compiler, Cmake version >= 2.8, BLAS & LAPACK, (MPI)
    - Download Trilinos: trilinos.org -> Download
  - Find a configuration script suitable for your computer
    - https://code.google.com/p/trilinos/wiki/BuildScript
    - Trilinos/sampleScripts/
  - Modify the script if necessary, & use it to run CMake
  - make –jN, make –jN install
  - Build your programs against Trilinos
    - Use CMake with FIND_PACKAGE(Trilinos …), or
    - Use Make with Trilinos Makefile.export system