Scientific Software Ecosystems

Michael A. Heroux
Sandia National Laboratories

http://www.users.csbsju.edu/~mheroux/SciSwEcosystems.pdf

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.

Sandia National Laboratories

Goals for this presentation

- Motivate the need for and value of reusable scientific software.
- Understand the sparse linear algebra ecosystem.
- Look ahead to next-generation systems.
- Discuss productivity.
- Take any questions you have.

Quiz (True/False)

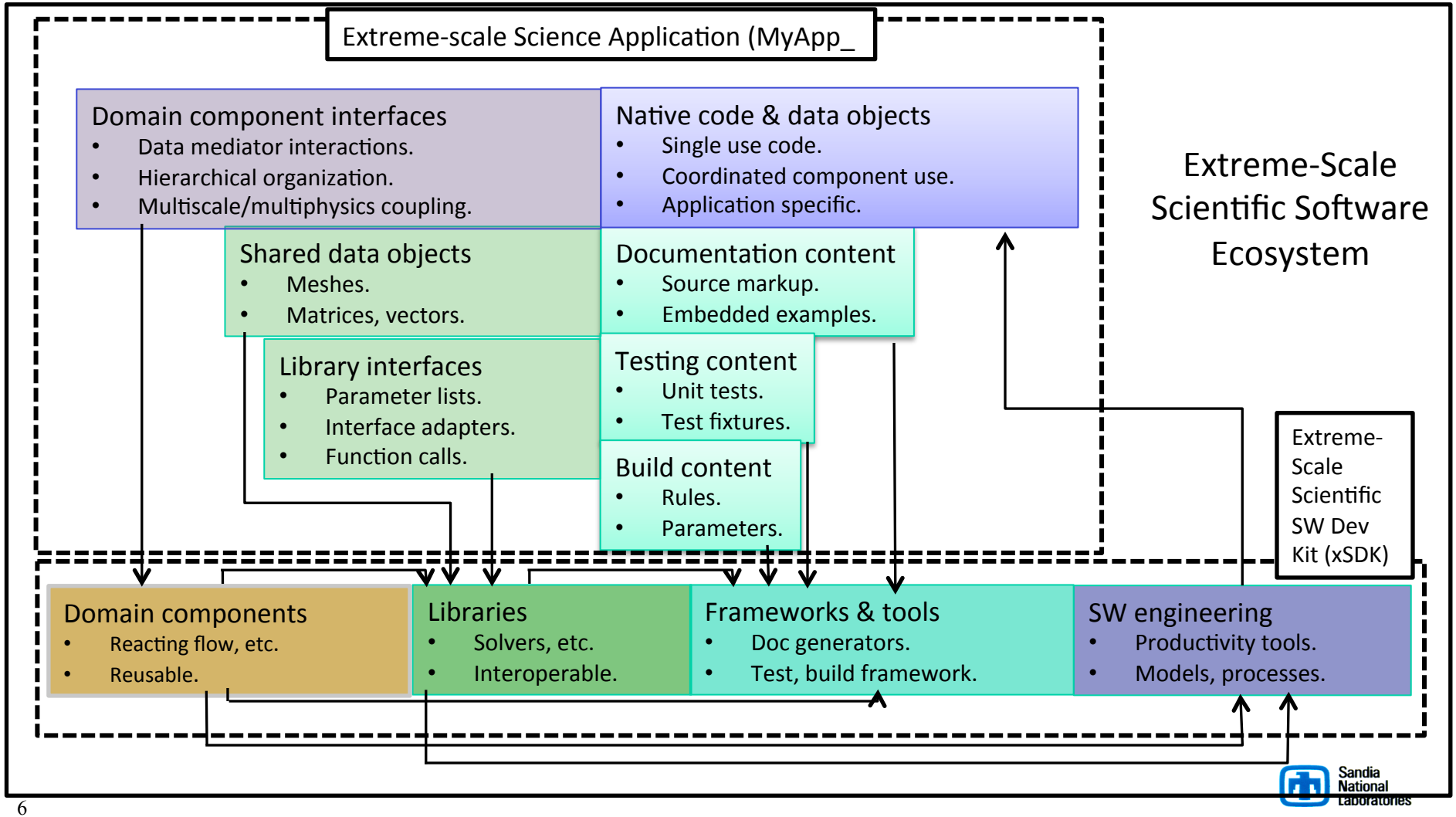
1. Building an app via reusable SW components is always a good idea.
2. Use of third party solvers is always a good idea.
3. Control inversion is a means of customizing SW ecosystem behavior.
4. Framework use must be an “all-in” commitment.
5. Performance portability requires data structure abstractions.
6. Operator abstractions enable sophisticated solution algorithms.
7. Algorithm development for massive concurrency is the “easy” part of our job.
8. Code optimization is always a good idea for HPC software.
9. Containerization capabilities are important for scientific SW.
10. Productivity must be a first-order, explicit focus for future scientific SW.

Basic Concepts

- Framework:
  - APIs, working software (defaults).
  - Control inversion, extensibility.
  - Scope: Big, ubiquitous.
- Toolkit:
  - “Plug-and-play” libraries, insertable.
  - Scope: Small, local.
- Lightweight framework: Goal is best of framework/toolkit.
- Ecosystem: Everything.

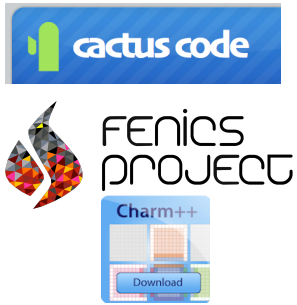
Modern Scientific App Design Goal

- Classic approach: *Develop* an application.
  - App has its own framework, no reuse intended.
  - Makes some use of libraries (toolkit components).
- Desired approach: *Compose* application within ecosystem.
  - Adapt lightweight framework elements.
    - For example: Use CMake, Doxygen, UnitTest frameworks.
  - Integrate & tune libraries.
    - Load balancing, solvers, etc.



Some Popular Ecosystems (Frameworks)

- Cactus: http://cactuscode.org
- FEniCS: http://fenicsproject.org/index.html
- Charm++: http://charm.cs.uiuc.edu
- PETSc: https://www.mcs.anl.gov/petsc/ (Will say more)



Trilinos Overview

What is Trilinos?

- 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

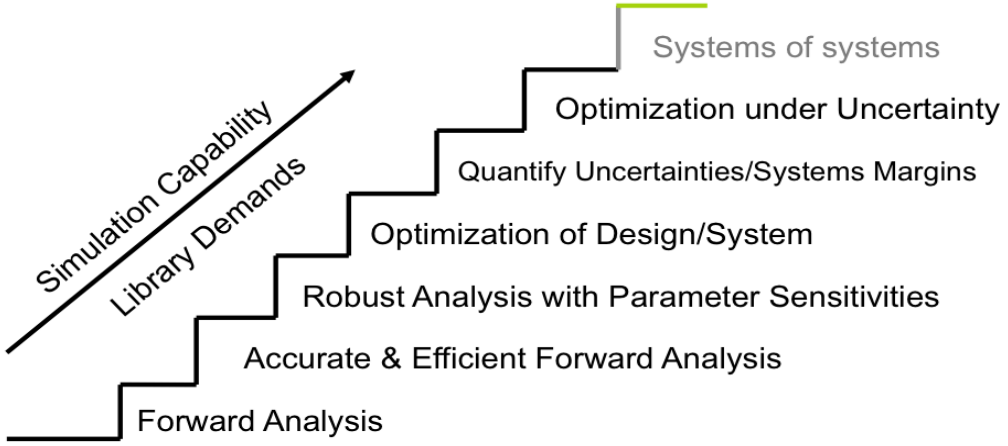
Laptops to
Leadership systems

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.

Capability Leaders: Layer of Proactive Leadership

- Areas:
  - Framework, Tools & Interfaces (J. Willenbring).
  - Software Engineering Technologies and Integration (R. Bartlett).
  - Discretizations (M. Perego).
  - 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

- 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…
- Support for mixed and arbitrary precisions
  - Don’t have to rebuild Trilinos to use it
- Support for huge (> 2B unknowns) problems

Trilinos Current Release

- Trilinos 12.4 current.
  - 57 packages.
  - But most people clone/fork directly from GitHub.com
- Website: https://trilinos.org .

Trilinos software organization

Trilinos Package Summary

	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.
  - AztecOO 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.

Trilinos is made of packages

- Not a monolithic piece of software
  - Like LEGO™ bricks, not Matlab™
- 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
- Trilinos is not “indivisible”
  - You don’t need all of Trilinos to get things done
  - Any subset of packages can be combined and distributed
  - Current public release contains ~50 of the 55+ Trilinos packages
- Trilinos top layer framework
  - Not a large amount of source code: ~1.5%
  - Manages package dependencies
    - Like a GNU/Linux package manager
  - Runs packages’ tests nightly, and on every check-in
- Package model supports multifrontal development
- New effort to create apps by gluing Trilinos together: Albany

Solver Software Stack

Phase I packages: SPMD, int/double

Phase II packages: Templated

Trilinos

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 $F(x) \in \Re^m \rightarrow$ Solve $F(x) = 0$ $x \in \Re^n$	NOX
Linear Problems Linear Equations: Eigen Problems:	Given Linear Ops (Matrices) $A, B \in \Re^{m \times$ Solve $Ax = b$ for $x \in \Re^n$ Solve $A\nu = \lambda B\nu$ for (all) $\nu \in \Re^n$, $\lambda \in$	Anasazi fpack, 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

Trilinos

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. $f(x,u) = 0$	Sensitivities (Automatic Differentiation: Sacado)	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 $F(x) \in \Re^m \rightarrow$ Solve $F(x) = 0$ $x \in \Re^n$		NOX | T-NOX
Linear Problems Linear Equations: Eigen Problems:	Given Linear Ops (Matrices) $A, B \in \Re^{m \times$ 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

Sparse Linear Systems

Sparse Matrices

- ***Sparse Matrix*** *(defn):* (not rigorous) An ***m***-by-***n*** matrix *with enough zero entries that it makes sense to keep track of what is zero and nonzero.*
- Example:

$$A = \begin{pmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{pmatrix}$$

Origins of Sparse Matrices

- In practice, *most* large matrices are sparse. Specific sources:
  - Differential equations.
    - Encompasses the vast majority of scientific and engineering simulation.
    - E.g., structural mechanics.
      - $F = ma$. Car crash simulation.
  - Stochastic processes.
    - Matrices describe probability distribution functions.
  - Networks.
    - Electrical and telecommunications networks.
    - Matrix element a_{ij} is nonzero if there is a wire connecting point i to point j.
  - 3D imagery for Google Earth
    - Relies on SuiteSparse (via the Ceres nonlinear least squares solver developed by Google).
  - And more…

Sparse Linear Systems: Problem Definition

- A frequent requirement for scientific and engineering computing is to solve:

$$Ax = b$$

where A is a known large (sparse) matrix ***a linear operator***,

b is a known vector,

x is an unknown vector.

Goal: Find x.

Why And How To Use Sparse Solver 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:

$$\begin{aligned} x + y &= 60 \\ 2x + 4y &= 200 \end{aligned}$$

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

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

- Solve for y:

$$\begin{aligned} 120 - 2y + 4y &= 200 \\ 2y &= 80 \\ y &= 40 \end{aligned}$$

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

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 Sparse Solver Libraries?

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

Importance of Sparse Solver 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.
  - Modern software design and architecture.
  - Large and growing support for manycore and accelerator devices.

Sparse Direct Methods

- Construct L and U, lower and upper triangular, resp, s.t.

$$LU = A$$

- Solve $Ax = b$:

1. $Ly = b$
2. $Ux = y$

- Symmetric versions: $LL^T = A$, LDL^T

- When are direct methods effective?
  - 1D: Always, even on many, many processors.
  - 2D: Almost always, except on many, many processors.
  - 2.5D: Most of the time.
  - 3D: Only for "small/medium" problems on "small/medium" processor counts.
- Bottom line: Direct sparse solvers should always be in your toolbox.

Sparse Direct Solver Packages

- HSL: http://www.hsl.rl.ac.uk
- MUMPS: http://mumps.enseeiht.fr
- Pardiso: http://www.pardiso-project.org
- PaStiX: http://pastix.gforge.inria.fr
- SuiteSparse: http://www.cise.ufl.edu/research/sparse/SuiteSparse
- SuperLU: http://crd-legacy.lbl.gov/~xiaoye/SuperLU/index.html
- UMFPACK : http://www.cise.ufl.edu/research/sparse/umfpack/
- WSMP: http://researcher.watson.ibm.com/researcher/view_project.php?id=1426
- Trilinos/Amesos/Amesos2: http://trilinos.org
- Notes:
  - All have threaded parallelism.
  - All but SuiteSparse and UMFPACK have distributed memory (MPI) parallelism.
  - MUMPS, PaStiX, SuiteSparse, SuperLU, Trilinos, UMFPACK are freely available.
  - HSL, Pardiso, WSMP are available freely, with restrictions.
  - Some research efforts on GPUs, unaware of any products.
- Emerging hybrid packages:
  - PDSLin – Sherry Li.
  - HIPS – Gaidamour, Henon.
  - Trilinos/ShyLU – Rajamanickam, Boman, Heroux.

Other Sparse Direct Solver Packages

- "Legagy" packages that are open source but not under active development today.
  - TAUCS : http://www.tau.ac.il/~stoledo/taucs/
  - PSPASES : http://www-users.cs.umn.edu/~mjoshi/pspases/
  - BCSLib : http://www.boeing.com/phantom/bcslib/

- Eigen http://eigen.tuxfamily.org
  - Newer, active, but sequential only (for sparse solvers).
  - Sparse Cholesky (including LDL^T), Sparse LU, Sparse QR.
  - Wrappers to quite a few third-party sparse direct solvers.

Emerging Trend in Sparse Direct

- New work in low-rank approximations to off-diagonal blocks.
- Typically:
  - Off-diagonal blocks in the factorization stored as dense matrices.
- New:
  - These blocks have low rank (up to the accuracy needed for solution).
  - Can be represented by approximate SVD.
- Still uncertain how broad the impact will be.
  - Will rank-*k* SVD continue to have low rank for hard problems?
- Potential: Could be breakthrough for extending sparse direct method to much larger 3D problems.

Iterative Methods

- Given an initial guess for x, called $x^{(0)}$, ($x^{(0)} = 0$ is acceptable) compute a sequence $x^{(k)}$, $k = 1,2, \ldots$ such that each $x^{(k)}$ is "closer" to x.
- Definition of "close":
  - Suppose $x^{(k)} = x$ exactly for some value of k.
  - Then $r^{(k)} = b - Ax^{(k)} = 0$ (the vector of all zeros).
  - And $norm(r^{(k)}) = sqrt(<r^{(k)}, r^{(k)}>) = 0$ (a number).
  - For any $x^{(k)}$, let $r^{(k)} = b - Ax^{(k)}$
  - If $norm(r^{(k)}) = sqrt(<r^{(k)}, r^{(k)}>)$ is small (< 1.0E-6 say) then we say that $x^{(k)}$ is close to x.
  - The vector r is called the residual vector.

What is preconditioning?

Denote the linear system (user's problem): $Ax = b$

Preconditioning: given M such that: $M = M_1 M_2$

Solve the preconditioned system: $\tilde{A}\tilde{x} = \tilde{b}$

Where:

$$\tilde{A} = M_1^{-1} A M_2^{-1}$$
$$\tilde{b} = M_1^{-1} b$$
$$\tilde{x} = M_2 x$$

The art of preconditioning:

- $\tilde{A}$ is close to identity, or nice properties
- M Inverse can be applied efficiently

Sparse Iterative Solver Packages

- PETSc: http://www.mcs.anl.gov/petsc
- hypre: https://computation.llnl.gov/casc/linear_solvers/sls_hypre.html
- Trilinos: http://trilinos.sandia.gov
- Paralution: http://www.paralution.com (Manycore; GPL/Commercial license)
- HSL: http://www.hsl.rl.ac.uk (Academic/Commercial License)
- Eigen http://eigen.tuxfamily.org (Sequential CG, BiCGSTAB, ILUT/Sparskit)
- Sparskit: http://www-users.cs.umn.edu/~saad/software
- Notes:
  - There are many other efforts, but I am unaware of any that have a broad user base like hypre, PETSc and Trilinos.
  - Sparskit, and other software by Yousef Saad, is not a product with a large official user base, but these codes appear as embedded (serial) source code in many applications.
  - PETSc and Trilinos support threading, distributed memory (MPI) and growing functionality for accelerators.
  - Many of the direct solver packages support some kind of iteration, if only iterative refinement.

Which Type of Solver to Use?

Dimension	Type	Notes
1D	Direct	Often tridiagonal (Thomas alg, periodic version).
2D *very* easy	Iterative	If you have a good initial guess, e.g., transient simulation.
2D otherwise	Direct	Almost always better than iterative.
2.5D	Direct	Example: shell problems. Good ordering can keep fill low.
3D "smooth"	Direct?	Emerging methods for low-rank SVD representation.
3D easy	Iterative	Simple preconditioners: diagonal scaling. CG or BiCGSTAB.
3D harder	Iterative	Prec: IC, ILU (with domain decomposition if in parallel).
3D hard	Iterative	Use GMRES (without restart if possible).
3D + large	Iterative	Add multigrid, geometric or algebraic.

Using Trilinos Linear Solvers

Sandia National Laboratories

Trilinos Package Summary

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

A Simple Epetra/AztecOO Program

```
// Header files omitted…
int main(int argc, char *argv[]) {
Epetra_SerialComm Comm();
```

```
// ***** Map puts same number of equations on each pe *****
  int NumMyElements = 1000 ;
  Epetra_Map Map(-1, NumMyElements, 0, Comm);
  int NumGlobalElements = Map.NumGlobalElements();
```

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

```
// ***** Create Linear Problem *****
  Epetra_LinearProblem problem(&A, &x, &b);
```

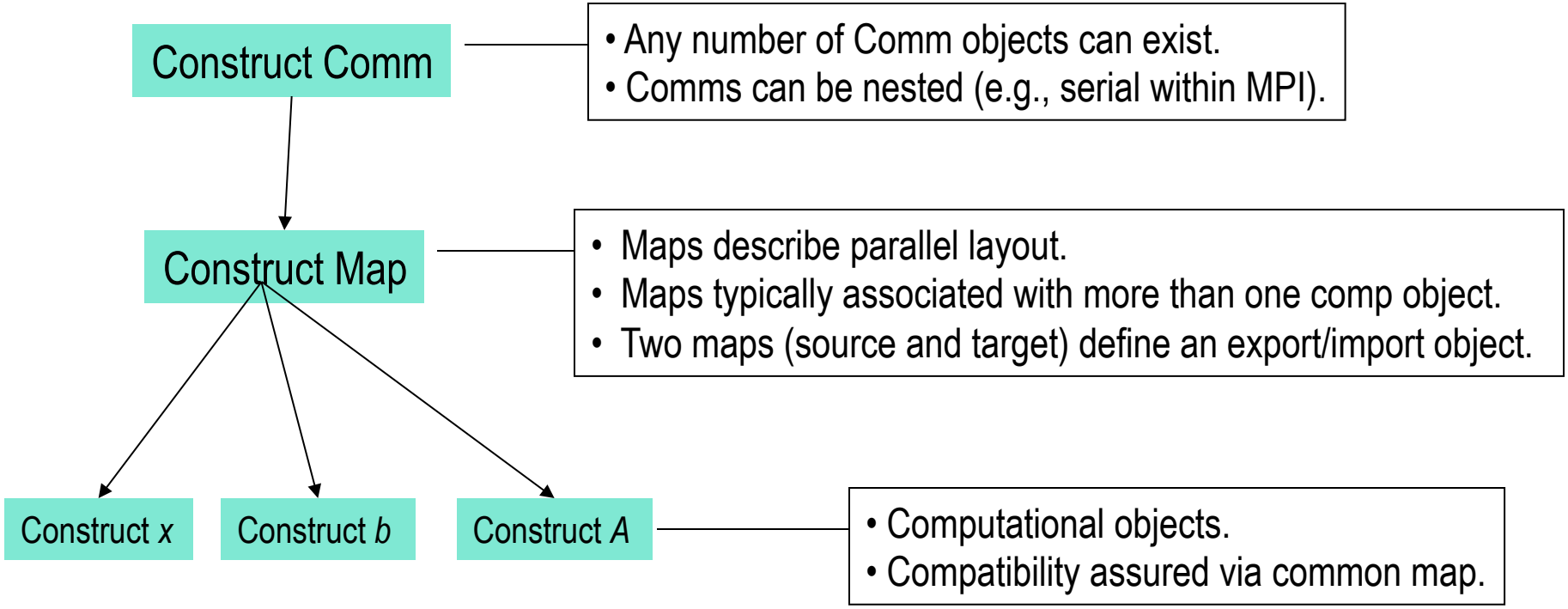
```
  // ***** Create/define AztecOO instance, solve *****
  AztecOO solver(problem);
  Teuchos::ParameterList parameterlist;
  parameterlist.set("precond", "Jacobi");
  solver.SetParameters(parameterlist);
  solver.Iterate(1000, 1.0E-8);
```

```
// ***** Report results, finish **********************
  cout << "Solver performed " << solver.NumIters()
       << " iterations." << endl
       << "Norm of true residual = "
       << solver.TrueResidual()
       << endl;


  return 0;
}
```

Sandia National Laboratories

Solving *Ax = b:*

Typical Petra Object Construction Sequence



Petra Implementations

- Epetra (Essential Petra):
  - Legacy production version
  - Uses stable core subset of C++ (circa 2000)
  - Restricted to real, double precision arithmetic
  - Interfaces accessible to C and Fortran users

- Tpetra (Templated Petra):
  - Next-generation version
  - C++ compiler must be C++11 compliant.
  - 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 manycore node library

Trilinos Package Summary

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

AztecOO

- Iterative linear solvers: CG, GMRES, BiCGSTAB,…
- Incomplete factorization preconditioners

- Aztec was Sandia's workhorse solver:
  - Extracted from the MPSalsa reacting flow code
  - Installed in dozens of Sandia apps
  - 1900+ external licenses
- AztecOO improves on Aztec by:
  - Using Epetra objects for defining matrix and vectors
  - Providing more preconditioners & scalings
  - Using C++ class design to enable more sophisticated use
- AztecOO interface allows:
  - Continued use of Aztec for functionality
  - Introduction of new solver capabilities outside of Aztec

Belos

- Next-generation linear iterative solvers

- Decouples algorithms from linear algebra objects
  - Linear algebra library has full control over data layout and kernels
  - Improvement over AztecOO, which controlled vector & matrix layout
  - Essential for hybrid (MPI+X) parallelism
- Solves problems that apps really want to solve, faster:
  - Multiple right-hand sides: AX=B
  - Sequences of related systems: $(A + \Delta A_k) X_k = B + \Delta B_k$
- Many advanced methods for these types of systems
  - Block & pseudoblock solvers: GMRES & CG
  - Recycling solvers: GCRODR (GMRES) & CG
  - “Seed” solvers (hybrid GMRES)
  - Block orthogonalizations (TSQR)
- Supports arbitrary & mixed precision, complex, …
- If you have a choice, pick Belos over AztecOO

Ifpack(2): Algebraic preconditioners

- Preconditioners:
  - Overlapping domain decomposition
  - Incomplete factorizations (within an MPI process)
  - (Block) relaxations & Chebyshev
- Accepts user matrix via abstract matrix interface
- Use {E,T}petra for basic matrix / vector calculations
- Perturbation stabilizations & condition estimation
- Can be used by all other Trilinos solver packages
- Ifpack2: Tpetra version of Ifpack
  - Supports arbitrary precision & complex arithmetic
  - Path forward to hybrid-parallel factorizations

: Multi-level Preconditioners

- Smoothed aggregation, multigrid, & domain decomposition

- Critical technology for scalable performance of many apps
- ML compatible with other Trilinos packages:
  - Accepts Epetra sparse matrices & dense vectors
  - ML preconditioners can be used by AztecOO, Belos, & Anasazi
- Can also be used independent of other Trilinos packages
- Next-generation version of ML: MueLu
  - Works with Epetra or Tpetra objects (via Xpetra interface)

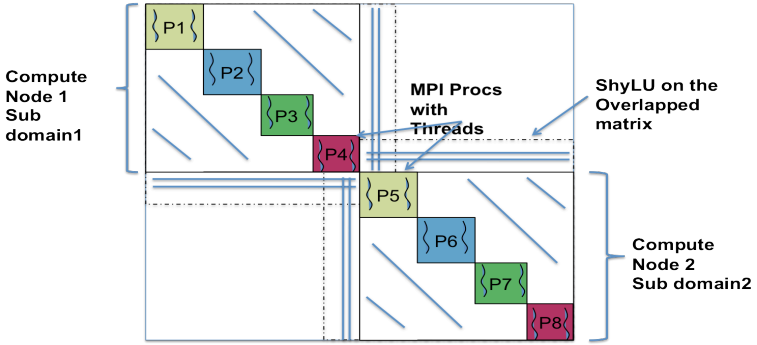
MueLu: Next-gen algebraic multigrid

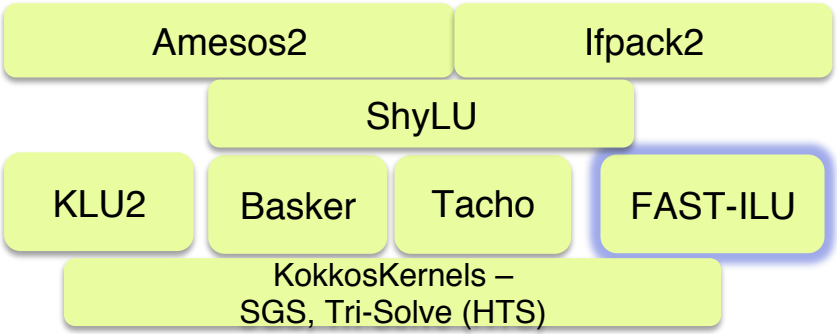
- Motivation for replacing ML
  - Improve maintainability & ease development of new algorithms
  - Decouple computational kernels from algorithms
    - ML mostly monolithic (& 50K lines of code)
    - MueLu relies more on other Trilinos packages
  - Exploit Tpetra features
    - MPI+X (Kokkos programming model mitigates risk)
    - 64-bit global indices (to solve problems with >2B unknowns)
    - Arbitrary Scalar types (Tramonto runs MueLu w/ double-double)
- Works with Epetra or Tpetra (via Xpetra common interface)
- Facilitate algorithm development
  - Energy minimization methods
  - Geometric or classic algebraic multigrid; mix methods together
- Better support for preconditioner reuse
  - Explore options between “blow it away” & reuse without change

Amesos/Amesos2

- Direct Solver interface for the Epetra/Tpetra Stack.
- Typical Usage:
  - *preOrder(),*
  - *symbolicFactorization(),*
  - *numericFactorization(),*
  - *solve().*
- Easy to support new solvers (Current support for all the SuperLU variants).
- Easy to support new multivectors and sparse matrices.
- Can support third party solver specific parameters with little changes.
- Available in the current release of Trilinos.

ShyLU and Subdomain Solvers : Overview





- MPI+X based subdomain solvers
  - Decouple the notion of one MPI rank as one subdomain: Subdomains can span multiple MPI ranks each with its own subdomain solver using X or MPI+X
  - Epetra based solver, Tpetra interface still being developed
    - Trilinos Solver Factory a big step forward to get this done (*Thanks to M. Hoemmen*)
- **Subpackages of ShyLU**: Multiple Kokkos-based options for on-node parallelism
  - **Basker :** LU or ILU (t) factorization (J. Booth)
  - **Tacho:** Incomplete Cholesky - IC (k) (K. Kim)
  - **Fast-ILU:** Fast-ILU factorization for GPUs (A. Patel)
- **KokkosKernels:** Coloring based Gauss-Seidel (M. Deveci), Triangular Solves
- Experimental code base under active development.

Abstract solver interfaces & applications

Stratimikos package

- Greek **στρατηγική** (strategy) + **γραμμικός** (linear)
- Uniform run-time interface to many different packages'
  - Linear solvers: Amesos, AztecOO, Belos, …
  - Preconditioners: Ifpack, ML, …
- Defines common interface to create and use linear solvers
- Reads in options through a Teuchos::ParameterList
  - Can change solver and its options at run time
  - Can validate options, & read them from a string or XML file
- Accepts any linear system objects that provide
  - Epetra_Operator / Epetra_RowMatrix view of the matrix
  - Vector views (e.g., Epetra_MultiVector) for right-hand side and initial guess
- Increasing support for Tpetra objects

Stratimikos Parameter List and Sublists

```
<ParameterList name="Stratimikos">
  <Parameter name="Linear Solver Type" type="string" value="AztecOO"/>
  <Parameter name="Preconditioner Type" type="string" value="Ifpack"/>
  <ParameterList name="Linear Solver Types">
    <ParameterList name="Amesos">
      <Parameter name="Solver Type" type="string" value="Klu"/>
      <ParameterList name="Amesos Settings">
        <Parameter name="MatrixProperty" type="string" value="general"/>
        ...
        <ParameterList name="Mumps"> ... </ParameterList>
        <ParameterList name="Superludist"> ... </ParameterList>
      </ParameterList>
    </ParameterList>
    <ParameterList name="AztecOO">
      <ParameterList name="Forward Solve">
        <Parameter name="Max Iterations" type="int" value="400"/>
        <Parameter name="Tolerance" type="double" value="1e-06"/>
        <ParameterList name="AztecOO Settings">
          <Parameter name="Aztec Solver" type="string" value="GMRES"/>
          ...
        </ParameterList>
      </ParameterList>
      ...
    </ParameterList>
    <ParameterList name="Belos"> ... (Details omitted) ... </ParameterList>
  </ParameterList>
<ParameterList name="Preconditioner Types">
    <ParameterList name="Ifpack">
      <Parameter name="Prec Type" type="string" value="ILU"/>
      <Parameter name="Overlap" type="int" value="0"/>
      <ParameterList name="Ifpack Settings">
        <Parameter name="fact: level-of-fill" type="int" value="0"/>
        ...
      </ParameterList>
    </ParameterList>
    <ParameterList name="ML"> ... (Details omitted) ... </ParameterList>
  </ParameterList>
</ParameterList>
```

Top level parameters

Linear Solvers

Preconditioners

Sublists passed on to package code.

Every parameter and sublist is handled by Thyra code and is fully validated.

Stratimikos Parameter List and Sublists

```
<ParameterList name="Stratimikos">
  <Parameter name="Linear Solver Type" type="string" value="Belos"/>
  <Parameter name="Preconditioner Type" type="string" value="ML"/>
  <ParameterList name="Linear Solver Types">
    <ParameterList name="Amesos">
      <Parameter name="Solver Type" type="string" value="Klu"/>
      <ParameterList name="Amesos Settings">
        <Parameter name="MatrixProperty" type="string" value="general"/>
        ...
        <ParameterList name="Mumps"> ... </ParameterList>
        <ParameterList name="Superludist"> ... </ParameterList>
      </ParameterList>
    </ParameterList>
    <ParameterList name="AztecOO">
      <ParameterList name="Forward Solve">
        <Parameter name="Max Iterations" type="int" value="400"/>
        <Parameter name="Tolerance" type="double" value="1e-06"/>
        <ParameterList name="AztecOO Settings">
          <Parameter name="Aztec Solver" type="string" value="GMRES"/>
          ...
        </ParameterList>
      </ParameterList>
      ...
    </ParameterList>
    <ParameterList name="Belos"> ... (Details omitted) ... </ParameterList>
  </ParameterList>
<ParameterList name="Preconditioner Types">
    <ParameterList name="Ifpack">
      <Parameter name="Prec Type" type="string" value="ILU"/>
      <Parameter name="Overlap" type="int" value="0"/>
      <ParameterList name="Ifpack Settings">
        <Parameter name="fact: level-of-fill" type="int" value="0"/>
        ...
      </ParameterList>
    </ParameterList>
    <ParameterList name="ML"> ... (Details omitted) ... </ParameterList>
  </ParameterList>
</ParameterList>
```

Top level parameters

Linear Solvers

Preconditioners

Solver/ preconditioner changed by single argument.

Parameter list is standard XML. Can be read from command line, file, string or hand-coded.

Parameter List Validation

Error Messages for Improper Parameters/Sublists

Example: User misspells “Aztec Solver” as “ztec Solver”

```
<ParameterList>
  <Parameter name="Linear Solver Type" type="string" value="AztecOO"/>
  <ParameterList name="Linear Solver Types">
    <ParameterList name="AztecOO">
      <ParameterList name="Forward Solve">
        <ParameterList name="AztecOO Settings">
          <Parameter name="ztec Solver" type="string" value="GMRES"/>
        </ParameterList>
      </ParameterList>
    </ParameterList>
  </ParameterList>
</ParameterList>
```

Error message generated from PL::validateParameters(…) with exception:

```
  Error, the parameter {name="ztec Solver",type="string",value="GMRES"}
  in the parameter (sub)list "RealLinearSolverBuilder->Linear Solver Types->AztecOO->Forward Solve->AztecOO
Settings"
  was not found in the list of valid parameters!

  The valid parameters and types are:
       {
            "Aztec Preconditioner" : string = ilu
            "Aztec Solver" : string = GMRES
            …
       }
```

Error Messages for Improper Parameters/Sublists

Example: User specifies the wrong type for “Aztec Solver”

```
<ParameterList>
  <Parameter name="Linear Solver Type" type="string" value="AztecOO"/>
  <Parameter name="Preconditioner Type" type="string" value="Ifpack"/>
  <ParameterList name="Linear Solver Types">
    <ParameterList name="AztecOO">
      <ParameterList name="Forward Solve">
        <ParameterList name="AztecOO Settings">
          <Parameter name="Aztec Solver" type="int" value="GMRES"/>
        </ParameterList>
      </ParameterList>
    </ParameterList>
  </ParameterList>
</ParameterList>
```

Error message generated from PL::validateParameters(…) with exception:

```
  Error, the parameter {paramName="Aztec Solver",type="int"}
  in the sublist "DefaultRealLinearSolverBuilder->Linear Solver Types->AztecOO->Forward Solve->AztecOO
Settings"
  has the wrong type.  The correct type is "string"!
```

Error Messages for Improper Parameters/Sublists

Example: User specifies the wrong value for “Aztec Solver”

```
<ParameterList>
  <Parameter name="Linear Solver Type" type="string" value="AztecOO"/>
  <Parameter name="Preconditioner Type" type="string" value="Ifpack"/>
  <ParameterList name="Linear Solver Types">
    <ParameterList name="AztecOO">
      <ParameterList name="Forward Solve">
        <ParameterList name="AztecOO Settings">
          <Parameter name="Aztec Solver" type="string" value="GMRESS"/>
        </ParameterList>
      </ParameterList>
    </ParameterList>
  </ParameterList>
</ParameterList>
```

Error message generated from PL::validateParameters(...) with exception:

```
Error, the value "GMRESS" is not recognized for the parameter "Aztec Solver" in the sublist "".

Valid selections include: "CG", "GMRES", "CGS", "TFQMR", "BiCGStab", "LU".
```

Stratimikos Details

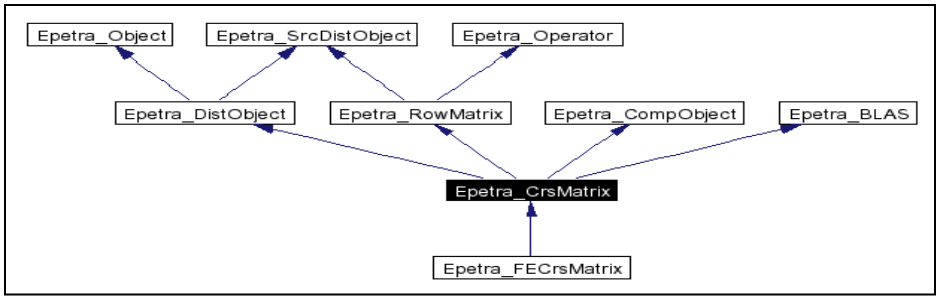
- Stratimikos has just one primary class:
  - Stratimikos::DefaultLinearSolverBuilder
  - An instance of this class accepts a parameter list that defines:
    - Linear Solver: Amesos, AztecOO, Belos.
    - Preconditioner: Ifpack, ML, AztecOO.
- Albany, other apps:
  - Access solvers through Stratimikos.
  - Parameter list is standard XML. Can be:
    - Read from command line.
    - Read from a file.
    - Passed in as a string.
    - Defined interactively.
    - Hand coded in source code.

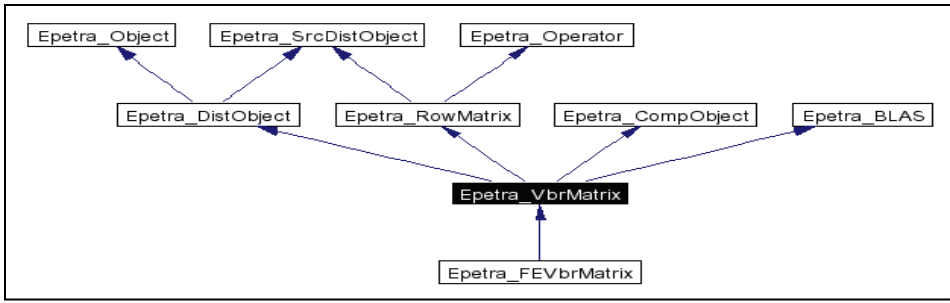
Combining Trilinos Packages To Solve Linear Systems

Tpetra/Epetra User Class Categories

- Sparse Matrices: ***RowMatrix***, (CrsMatrix, VbrMatrix, FECrsMatrix, …)
- Linear Operator: ***Operator*:** (Belos, AztecOO, ML Muelu, Ifpack/2, …)
- Vectors: Vector, MultiVector
- Graphs: CrsGraph
- Data Layout: Map, BlockMap, LocalMap
- Redistribution: Import, Export
- Aggregates: LinearProblem
- Parallel Machine: ***Comm***, (SerialComm, MpiComm, MpiSmpComm)

Matrix Class Inheritance Details





CrsMatrix and VbrMatrix inherit from:

- Distributed Object: How data is spread across machine.
- Computational Object: Performs FLOPS.
- BLAS: Use BLAS kernels.
- RowMatrix: An object from either class has a common row access interface.

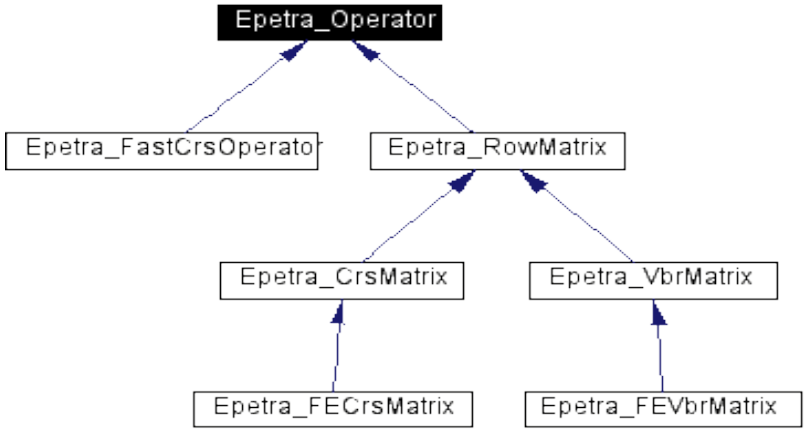
Sandia National Laboratories

Belos/AztecOO Extensibility

- Designed to accept externally defined:
  - Operators (both *A* and *M*):
    - The linear operator *A* is accessed as an Epetra_Operator.
    - Users can register a preconstructed preconditioner as an Epetra_Operator.
  - RowMatrix:
    - If *A* is registered as a RowMatrix, Trilinos preconditioners are accessible.
    - Alternatively *M* can be registered separately as an Epetra_RowMatrix, and Trilinos preconditioners are accessible.
  - StatusTests:
    - Standard stopping criteria are accessible.
    - Can override these mechanisms by registering a StatusTest Object.

Belos/AztecOO understands Epetra_Operator

- Designed to accept externally defined:
  - Operators (both *A* and *M*).
  - RowMatrix (Facilitates use of AztecOO preconditioners with external *A*).
  - StatusTests (externally-defined stopping criteria).



Sandia National Laboratories

A Simple Epetra/AztecOO Program

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

```
// ***** Map puts same number of equations on each pe *****
  int NumMyElements = 1000 ;
  Epetra_Map Map(-1, NumMyElements, 0, Comm);
  int NumGlobalElements = Map.NumGlobalElements();
```

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

```
// ***** Create Linear Problem *****
  Epetra_LinearProblem problem(&A, &x, &b);
```

```
  // ***** Create/define AztecOO instance, solve *****
  AztecOO solver(problem);
  Teuchos::ParameterList parameterlist;
  parameterlist.set("precond", "Jacobi");
  solver.SetParameters(parameterlist);
  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;
}
```

Sandia National Laboratories

AztecOO with ML

Trilinos/packages/aztecoo/example/MLAztecOO

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

```
// ***** Map puts same number of equations on each pe *****
  int NumMyElements = 1000 ;
  Epetra_Map Map(-1, NumMyElements, 0, Comm);
  int NumGlobalElements = Map.NumGlobalElements();
```

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

```
// ***** Create Linear Problem *****
  Epetra_LinearProblem problem(&A, &x, &b);
```

```
  // ***** Create/define AztecOO instance, solve *****
  AztecOO solver(problem);
 // Code from next slide gets inserted here.
  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;
}
```

Sandia National Laboratories

ML Preconditioner Setup

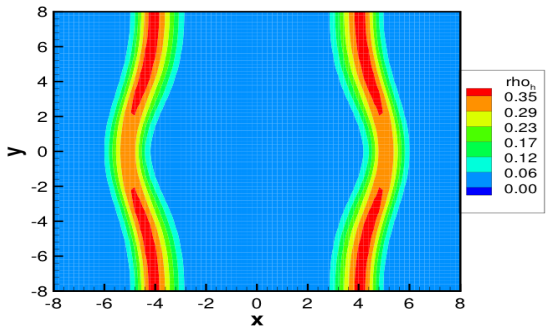
```
cout << "Creating multigrid hierarchy" << endl;
ML *ml_handle;
int N_levels = 10;

ML_Set_PrintLevel(3);

ML_Create(&ml_handle,N_levels);
EpetraMatrix2MLMatrix(ml_handle, 0, &A);

ML_Aggregate *agg_object;
ML_Aggregate_Create(&agg_object);
ML_Aggregate_Set_MaxCoarseSize(agg_object,1);
N_levels = ML_Gen_MGHierarchy_UsingAggregation(ml_handle, 0,
                                    ML_INCREASING, agg_object);
// Set a symmetric Gauss-Seidel smoother for the MG method
ML_Gen_Smoother_SymGaussSeidel(ml_handle, ML_ALL_LEVELS,
                           ML_BOTH, 1, ML_DEFAULT);
ML_Gen_Solver    (ml_handle, ML_MGV, 0, N_levels-1);
cout << "Creating Epetra_ML_Operator wrapper" << endl;
Epetra_ML_Operator  MLop(ml_handle,comm,map,map);
//note: SetPrecOperator() also sets AZ_user_precond
solver.SetPrecOperator(&MLop);
```

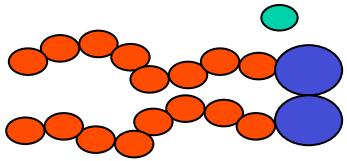
ML:

- Independent from AztecOO
- Implements Epetra_Operator interface
- Can be used with AztecOO.

Self-assembly of lipid bilayers Using Tramonto

https://software.sandia.gov/tramonto/





8-2-8 Chain

Sandia National Laboratories

CMS-DFT / polymers

- developed for polymers
- chains are flexible
- 2nd order density expansion

Chandler, McCoy, Singer (1986);
McCoy et al. (1990s)

$$\rho_\alpha(r) = \frac{\rho_\alpha^b}{N_\alpha} \sum_{s=1}^{N_\alpha} \frac{G_s(r) G_s^i(r)}{e^{-\beta U_\alpha(r)}}$$

$$U_\alpha(r) = V_{ext}(r) - \sum_\gamma \int c_{\alpha\gamma}(r - r')[\rho_\gamma(r') - \rho_\gamma^b]dr'$$

$$G_s(r) = e^{-\beta U_{\alpha,s}} \int w(r - r') G_{s-1}(r') dr'$$

$$G_s^i(r) = e^{-\beta U_{\alpha,s}} \int w(r - r') G_{s+1}^i(r') dr'$$

$$G_1 = G_N^i = e^{-\beta U(r)}$$

$$w(r) = \frac{1}{4\pi\sigma^2} \delta(|r| - \sigma)$$

Chain density distribution

Mean field

$$c(r) = c_{rep}(r) - u_{att}(r)$$

PRISM Theory **RPM Approx**

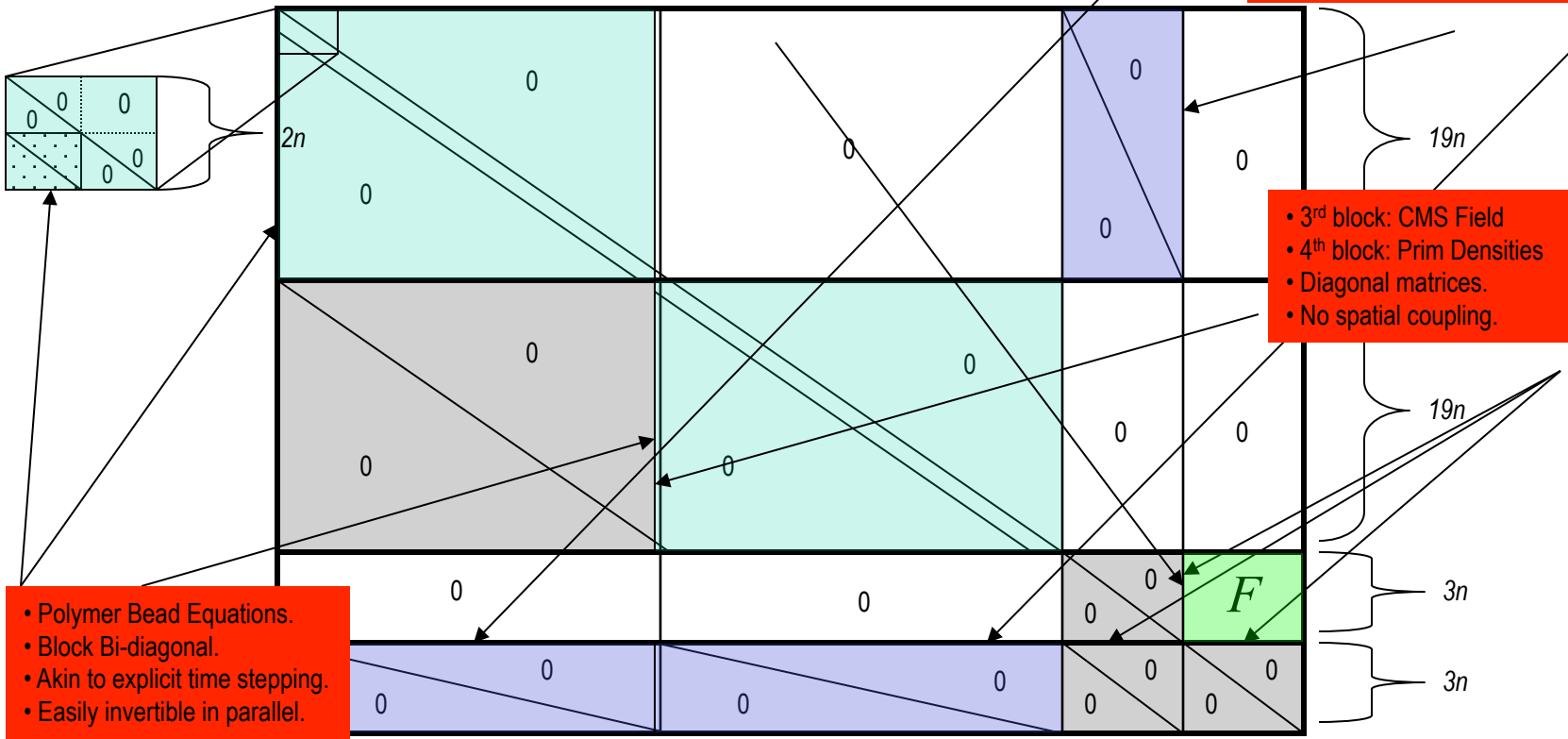
Chain Architecture (freely-jointed chains)

Sandia National Laboratories

- There is only ONE interesting block in this whole matrix.
- *F* describes CMS field dependence on primitive densities.
- 2.5σ radius integral at each grid node (mesh independent).
- Not sparse, nor dense. Constant coefficient.

lem

- Diagonal-like.
- One non-zero per row/col in long dimension.
- Like Prolongation/restriction Operators?



- 3rd block: CMS Field
- 4th block: Prim Densities
- Diagonal matrices.
- No spatial coupling.

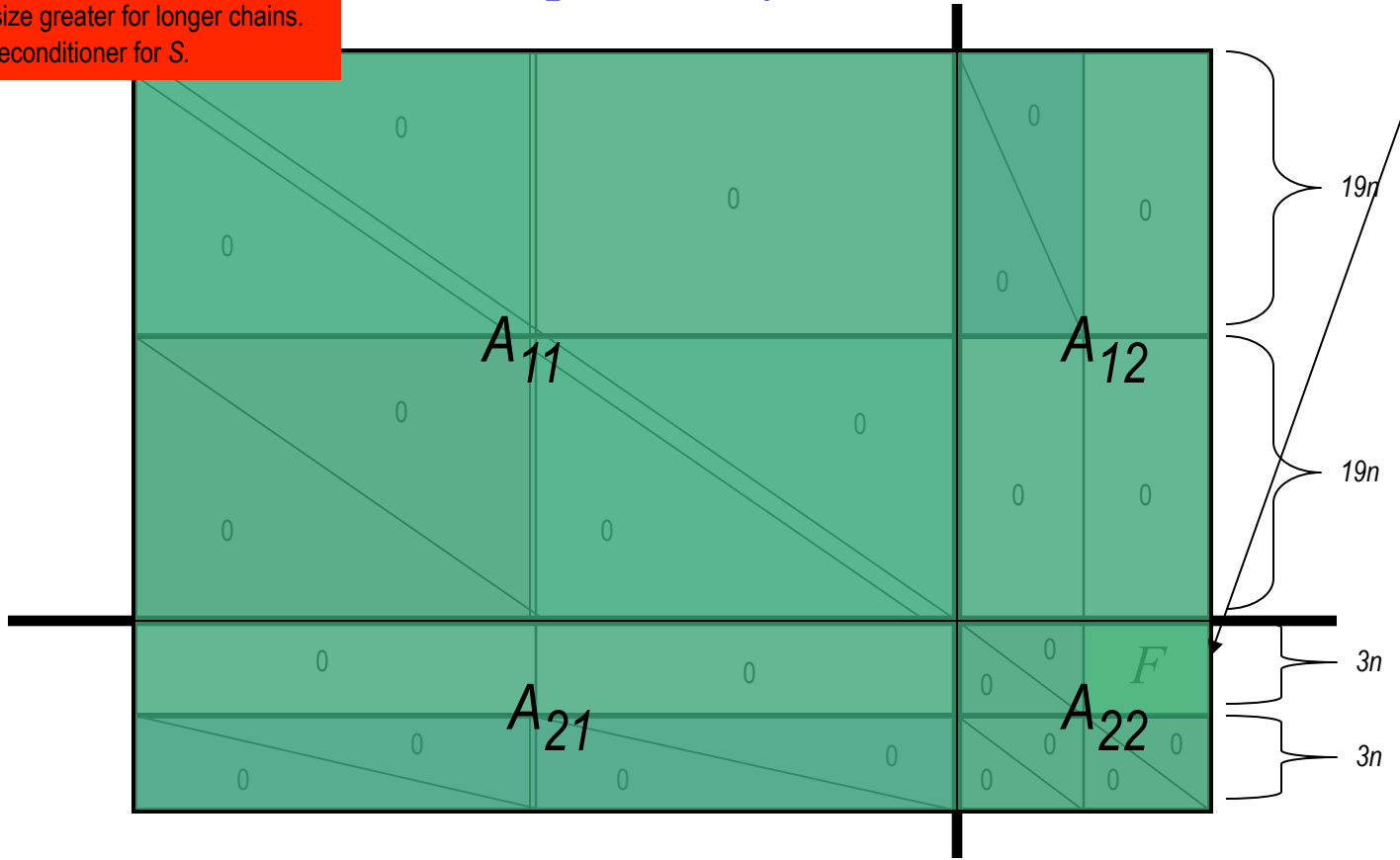
- Polymer Bead Equations.
- Block Bi-diagonal.
- Akin to explicit time stepping.
- Easily invertible in parallel.

Sandia National Laboratories

- Last layer of structure: 2-by-2 partitioning.
- A_{11} solve easily applied in parallel.
- Apply GMRES to $S = A_{22} - A_{21}*inv(A_{11})*A_{12}$
- GMRES sees 6.6x reduction in problem size.
- Reduction in size greater for longer chains.
- Still need a preconditioner for S.

- F has strong overlap:
  Distribute separate from rest of problem.

Lipid Bi-Layer Problem



Sandia National Laboratories

Preconditioner for *S*

$$A_{22} = \begin{array}{|c|c|}\hline D_{11} & F \\ \hline D_{21} & D_{22} \\ \hline \end{array}$$

$$A_{22} \approx A_{22}^{P} = \begin{array}{|c|c|}\hline D_{11} & F \\ \hline 0 & D_{22} \\ \hline \end{array}$$

- D_{11}, D_{22} = O(1), D_{21} = O(1e-10)
- Ignore D_{21} for preconditioning.
- *P(S)* requires
  - 2 diagonal scalings,
  - matvec with F.
- All distributed operations.

2-by-2 Schur Complement Operator

Source excerpt from dft_2x2_schur_epetra_operator.hpp

```
//! dft_2x2_schur_epetra_operator: An implementation of the Epetra_Operator class for
Tramonto Schur complements.
/*! Special 2-by-2 block operator for Tramonto polymer and explicit non-local density problems.
*/

class dft_2x2_schur_epetra_operator: public virtual Epetra_Operator {

 public:

  //@{ \name Constructors.
    //! Builds an implicit composite operator from a 2-by-2 block system

  dft_2x2_schur_epetra_operator(Epetra_Operator * A11inv, Epetra_Operator * A12,
                                Epetra_Operator * A21, Epetra_Operator * A22);
  //@}
```

Source excerpt from dft_2x2_schur_epetra_operator.cpp

```
//==============================================================================
dft_2x2_schur_epetra_operator::dft_2x2_schur_epetra_operator(Epetra_Operator * A11inv, Epetra_Operator * A12,
                                                             Epetra_Operator * A21, Epetra_Operator * A22)
  : A11inv_(A11inv),
    A12_(A12),
    A21_(A21),
    A22_(A22),
    Label_(0) {

  Label_ = "dft_2x2_schur_epetra_operator";
}
//==============================================================================
dft_2x2_schur_epetra_operator::~dft_2x2_schur_epetra_operator() { // Destructor
}
//==============================================================================
// Apply Schur Complement Operator
int dft_2x2_schur_epetra_operator::Apply(const Epetra_MultiVector& X, Epetra_MultiVector& Y) const {


  if (!X.Map().SameAs(OperatorDomainMap())){EPETRA_CHK_ERR(-1);}
  if (!Y.Map().SameAs(OperatorRangeMap())) {EPETRA_CHK_ERR(-2);}
  if (Y.NumVectors()!=X.NumVectors()) {EPETRA_CHK_ERR(-3);}

  // Apply (A22 - A21*inv(A11)*A12 to X

  Epetra_MultiVector Y1(A12_->RangeMap(), X.NumVectors());
  Epetra_MultiVector Y2(A21_->RangeMap(), X.NumVectors());

  A12_->Apply(X, Y1);
  A11inv_->Apply(Y1, Y1);
  A21_->Apply(Y1, Y2);
  A22_->Apply(X, Y);
  Y.Update(-1.0, Y2, 1.0);
  return(0);
}
```

Sandia National Laboratories

Part II:
The “Hard” Challenges for Next Generation HPC Applications and Systems

Michael A. Heroux
Sandia National Laboratories

Primary SNL Collaborators: Ross Bartlett, Erik Boman, Carter Edwards, James Elliot, Marc Gamell, Mark Hoemmen, Alicia Klinvex, Siva Rajamanickam, Keita Teranishi, Christian Trott, Jim Willenbring

IDEAS Project: Lois McInnes, David Bernholdt, David Moulton, Hans Johansen

Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin company, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.

Outline

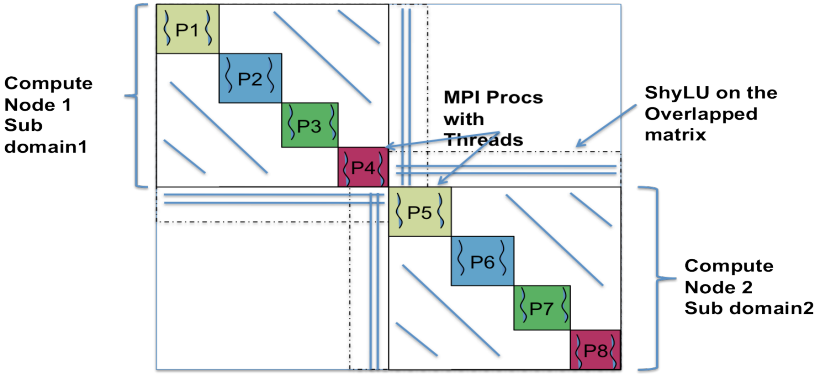
- “Easy” and “Hard”.
- SW Engineering and Productivity.
- Application Design and Productivity.
- Productivity Incentives.

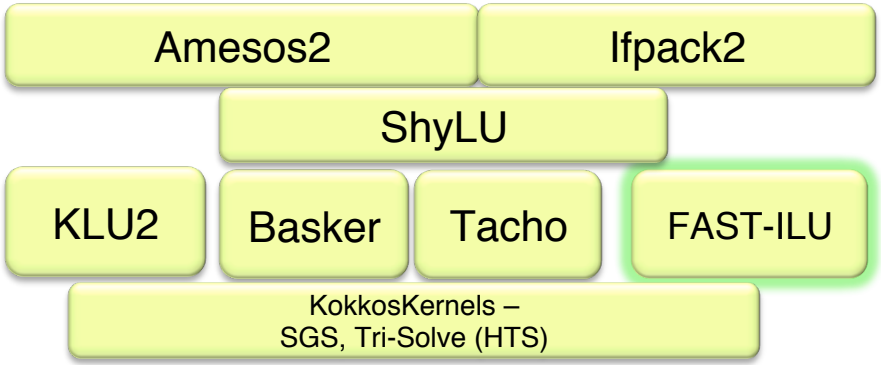
“Easy” Work in Progress

- Thread-scalable algorithms:
  - Turning out to be feasible.
  - Clever ideas: Fast-ILU (Chow, Anzt, Rajamanickam, etc.)
  - Lots to do, but steady progress
- Current Thread Programming Environments:
  - C++, OpenMP, others: Working.
  - Runtimes: Still a lot of work, but progress.
- Lots to do, but community is focused.

Trilinos/ShyLU and Subdomain Solvers : Overview

Led by Siva Rajamanickam, Sandia





- MPI+X based subdomain solvers
  - Decouple the notion of one MPI rank as one subdomain: Subdomains can span multiple MPI ranks each with its own subdomain solver using X or MPI+X
  - Epetra based solver, Tpetra interface still being developed
    - Trilinos Solver Factory a big step forward to get this done (*M. Hoemmen*)
- Subpackages of ShyLU: Multiple Kokkos-based options for on-node parallelism
  - Basker : LU or ILU (t) factorization (J. Booth)
  - Tacho: Incomplete Cholesky - IC (k) (K. Kim)
  - Fast-ILU: Fast-ILU factorization for GPUs (A. Patel)
- KokkosKernels: Coloring based Gauss-Seidel (M. Deveci), Triangular Solves
- **Experimental code base under active development.**

More “Easy” Work in Progress

- Resilience:
  - CPR: Compression, NVRAM, Offloading.
    - Steady progress, long life extension.
  - LFLR: Good progress with ULFM.
    - Example Paper: *Local Recovery And Failure Masking For Stencil-based Applications At Extreme Scales*
      - Marc Gamell, Keita Teranishi, Michael A. Heroux, Jackson Mayo, Hemanth Kolla, Jacqueline Chen, Manish Parashar
        http://sc15.supercomputing.org/schedule/event_detail?evid=pap682
- System-level error detection/correction.
  - Many unexploited options available. Talk with Al Gara, Intel.
- Conjecture:
  - *System developers will not permit reduced reliability until the user community produces more resilient apps.*

“Hard” Work

- Billions (yes, billions) SLOC of encoded science & engineering.
- Challenge:
  - Transfer, refactor, rewrite for modern systems.
  - Take advantage of disruption to re-architect.
  - Do so with modest investment bump up.
  - Deliver science at the same time.
  - Make the next disruption easier to address.

From the US NSCI Announcement (Fact sheet):

Improve HPC application developer productivity.

Current HPC systems are very difficult to program, requiring careful measurement and tuning to get maximum performance on the targeted machine. Shifting a program to a new machine can require repeating much of this process, and it also requires making sure the new code gets the same results as the old code. The level of expertise and effort required to develop HPC applications poses a major barrier to their widespread use.

Government agencies will support research on new approaches to building and programming HPC systems that make it possible to express programs at more abstract levels and then automatically map them onto specific machines. In working with vendors, agencies will emphasize the importance of programmer productivity as a design objective. Agencies will foster the transition of improved programming tools into actual practice, making the development of applications for HPC systems no more difficult than it is for other classes of large-scale systems.

https://www.whitehouse.gov/sites/default/files/microsites/ostp/nsci_fact_sheet.pdf

Productivity

Better, Faster, Cheaper: Pick all three

Confluence of trends

- Fundamental trends:
  - Disruptive HW changes: Requires thorough algorithm/code refactoring
  - Demands for coupling: Multiphysics, multiscale
- Challenges:
  - Need refactorings: Really, continuous change
  - Modest app development funding: No monolithic apps
  - Requirements are unfolding, evolving, not fully known *a priori*
- Opportunities:
  - Better design and SW practices & tools are available
  - Better SW architectures: Toolkits, libraries, frameworks
- Basic strategy: Focus on productivity

Interoperable Design of Extreme-scale Application Software (IDEAS)

IDEAS productivity

84

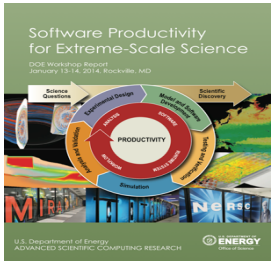
Motivation

Enable *increased scientific productivity,* realizing the potential of extreme- scale computing, through *a new interdisciplinary and agile approach to the scientific software ecosystem*.

Objectives

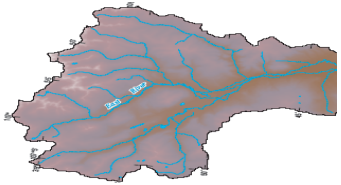
Address confluence of trends in hardware and increasing demands for predictive multiscale, multiphysics simulations.

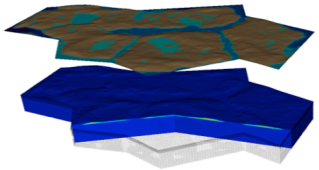
Respond to trend of continuous refactoring with efficient agile software engineering methodologies and improved software design.

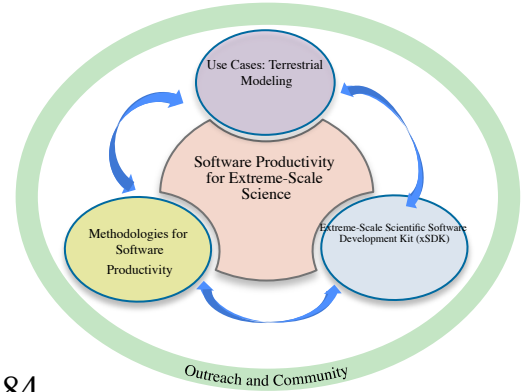


Impact on Applications & Programs

Terrestrial ecosystem *use cases tie IDEAS to modeling and simulation goals* in two Science Focus Area (SFA) programs and both Next Generation Ecosystem Experiment (NGEE) programs in DOE Biologic and Environmental Research (BER).







Approach

ASCR/BER partnership ensures delivery of both crosscutting methodologies and metrics with impact on real application and programs.

Interdisciplinary multi-lab team (ANL, LANL, LBNL, LLNL, ORNL, PNNL, SNL)

ASCR Co-Leads: Mike Heroux (SNL) and Lois Curfman McInnes (ANL)
BER Lead: David Moulton (LANL)
Topic Leads: David Bernholdt (ORNL) and Hans Johansen (LBNL)

Integration and synergistic advances in three communities deliver scientific productivity; outreach establishes a new holistic perspective for the broader scientific community.

www.ideas-productivity.org

U.S. DEPARTMENT OF ENERGY
Office of Science

IDEAS project structure and interactions

DOE Program Managers
ASCR: Thomas Ndousse-Fetter
BER: Paul Bayer, David Lesmes

IDEAS: Interoperable Design of Extreme-scale Application Software
ASCR Co-Leads: Mike Heroux (SNL) and Lois Curfman McInnes (ANL)
BER Lead: J. David Moulton (LANL)

Executive Advisory Board
John Cary (Tech-X)
Mike Glass (SNL)
Susan Hubbard (LBNL)
Doug Kothe (ORNL)
Sandy Landsberg (DOD)
Paul Messina (ANL)

BER Use Cases
Lead: J. David Moulton (LANL)
Carl Steefel (LBNL) *1
Scott Painter (ORNL) *2
Reed Maxwell (CSM) *3
Glenn Hammond (SNL)
Tim Scheibe (PNNL)
Laura Condon (CSM)
Ethan Coon (LANL)
Dipankar Dwivedi (LBNL)
Jeff Johnson (LBNL)
Eugene Kikinzon (LANL)
Sergi Molins (LBNL)
Steve Smith (LLNL)
Carol Woodward (LLNL)
Xiaofan Yang (PNNL)

Methodologies for Software Productivity
Lead: Mike Heroux (SNL)
Roscoe Bartlett (ORNL)
Todd Gamblin* (LLNL)
Christos Kartsaklis (ORNL)
Pat McCormick (LANL)
Sri Hari Krishna Narayanan (ANL)
Andrew Salinger* (SNL)
Jason Sarich (ANL)
Dali Wang (ORNL)
Jim Willenbring (SNL)

Extreme-Scale Scientific Software Development Kit
Lead: Lois Curfman McInnes (ANL)
Alicia Klinvex (SNL)
Jed Brown (ANL)
Irina Demeshko (SNL)
Anshu Dubey (LBNL)
Sherry Li (LBNL)
Vijay Mahadevan (ANL)
Daniel Osei-Kuffuor (LLNL)
Barry Smith (ANL)
Mathew Thomas (PNNL)
Ulrike Yang (LLNL)

Outreach and Community
Lead: David Bernholdt (ORNL)
Katie Antypas* (NERSC)
Lisa Childers* (ALCF)
Judy Hill* (OLCF)

IDEAS productivity

Crosscutting Lead: Hans Johansen (LBNL)

* Liaison
*1 *2 *3: Leads: Use Cases 1, 2, 3

Sept 2015

SFAs	NGEE	Exascale Co-Design	ASCR Math & CS	ALCF
CLM	ACME	Exascale Roadmap	SciDAC	NERSC	OLCF

BER Terrestrial Programs | DOE Extreme-scale Programs | DOE Computing Facilities

SW Engineering & Productivity

“A scientist builds in order to learn;
an engineer learns in order to build.”

- Fred Brooks

Scientist: Barely-sufficient building.

Engineer: Barely-sufficient learning.

Both: Insufficiency leads to poor SW.

Software Engineering and HPC: Efficiency vs Other Quality Metrics

How focusing on the factor below affects the factor to the right	Correctness	Usability	Efficiency	Reliability	Integrity	Adaptability	Accuracy	Robustness
Correctness	↑		↑	↑			↑	↓
Usability		↑				↑	↑	
Efficiency	↓		↑	↓	↓	↓	↓	
Reliability	↑			↑	↑		↑	↓
Integrity			↓	↑	↑			
Adaptability					↓	↑		↑
Accuracy	↑		↓	↑		↓	↑	↓
Robustness	↓	↑	↓	↓	↓	↑	↓	↑

Helps it ↑

Hurts it ↓

Source:
Code Complete
Steve McConnell

TriBITS: One Deliberate Approach to SE4CSE

Component-oriented SW Approach from Trilinos, CASL Projects, LifeV, …

Goal: "Self-sustaining" software

TriBITS Lifecycle Maturity Levels
0: Exploratory
1: Research Stable
2: Production Growth
3: Production Maintenance
-1: Unspecified Maturity

Goals

- *Allow Exploratory Research to Remain Productive*: Minimal practices for basic research in early phases
- *Enable Reproducible Research*: Minimal software quality aspects needed for producing credible research, researchers will produce better research that will stand a better chance of being published in quality journals that require reproducible research
- *Improve Overall Development Productivity*: Focus on the right SE practices at the right times, and the right priorities for a given phase/maturity level, developers work more productively with acceptable overhead
- *Improve Production Software Quality*: Focus on foundational issues first in early-phase development, higher-quality software will be produced as other elements of software quality are added
- *Better Communicate Maturity Levels with Customers*: Clearly define maturity levels so customers and stakeholders will have the right expectations

End of Life?

Long-term maintenance and end of life issues for Self-Sustaining Software:

- User community can help to maintain it (e.g., LAPACK).
- If the original development team is disbanded, users can take parts they are using and maintain it long term.
- Can stop being built and tested if not being currently used.
- However, if needed again, software can be resurrected, and continue to be maintained.

NOTE: Distributed version control using tools like Git greatly help in reducing risk and sustaining long lifetime.

Addressing existing Legacy Software

- One definition of "Legacy Software": Software that is too far from away from being Self-Sustaining Software, i.e:
  - Open-source
  - Core domain distillation document
  - Exceptionally well testing
  - Clean structure and code
  - Minimal controlled internal and external dependencies
  - Properties apply recursively to upstream software

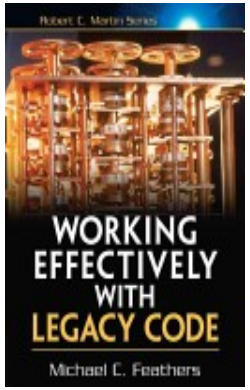
- Question: What about all the existing "Legacy" Software that we have to continue to develop and maintain? How does this lifecycle model apply to such software?

- Answer: Grandfather them into the TriBITS Lifecycle Model by applying the Legacy Software Change Algorithm.

Grandfathering of Existing Packages

Agile Legacy Software Change Algorithm:

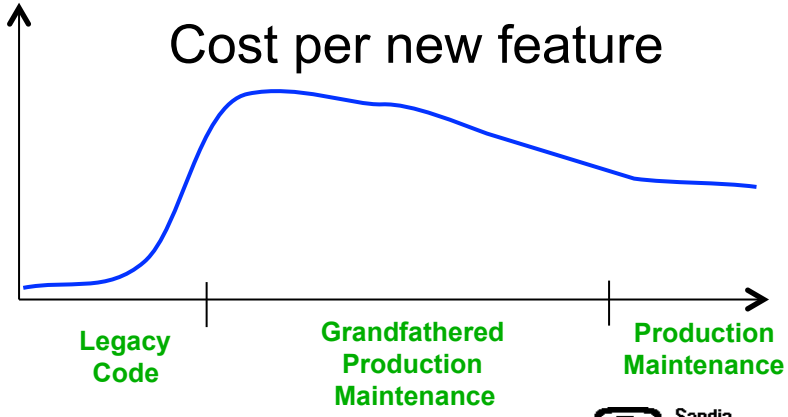
1. Identify Change Points
2. Break Dependencies
3. Cover with Unit Tests
4. Add New Functionality with Test Driven Development (TDD)
5. Refactor to removed duplication, clean up, etc.



Grandfathered Lifecycle Phases:

1. Grandfathered Research Stable (GRS) Code
2. Grandfathered Production Growth (GPG) Code
3. Grandfathered Production Maintenance (GPM) Code

NOTE: After enough iterations of the Legacy Software Change Algorithm the software may approach Self-Sustaining software and be able to remove the "Grandfathered" prefix.

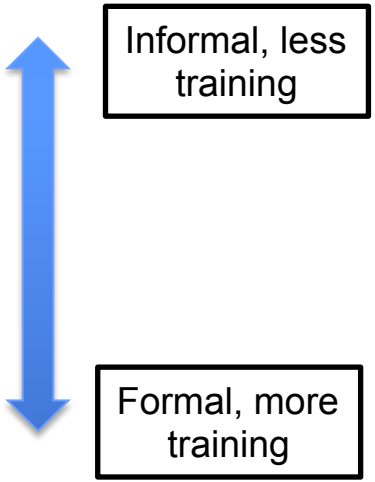


Message to This Audience

Write tests now, while (or before) writing your intended production software.

Continual Process Improvement Example: Managing issues

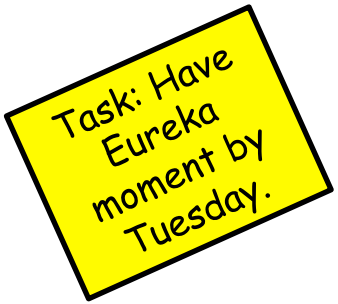
- Issue: Bug report, feature request
- Approaches:
  - Short-term memory, office notepad
  - ToDo.txt on computer desktop (1 person)
  - Issues.txt in repository root (small co-located team)
  - …
  - Web-based tool + Kanban (distributed, larger team)
  - Web-based tool + Scrum (full-time dev team)
- IDEAS project:
  - Jira Agile + Confluence: Turnkey web platform (ACME too)
  - Kanban: Simplest of widely known Agile SW dev processes



General Strategy: Assess your current processes, identify and execute improvement.
Always trying to improve your team or personal best practices.

Kanban principles

- Limit number of “In Progress” tasks
- Productivity improvement:
  - Optimize “flexibility vs swap overhead” balance. No overcommitting.
  - Productivity weakness exposed as bottleneck. Team must identify and fix the bottleneck.
  - Effective in R&D setting. Avoids a deadline-based approach. Deadlines are dealt with in a different way.
- Provides a board for viewing and managing issues



IDEAS Confluence, Jira Agile, Kanban

Pages / IDEAS Common Home

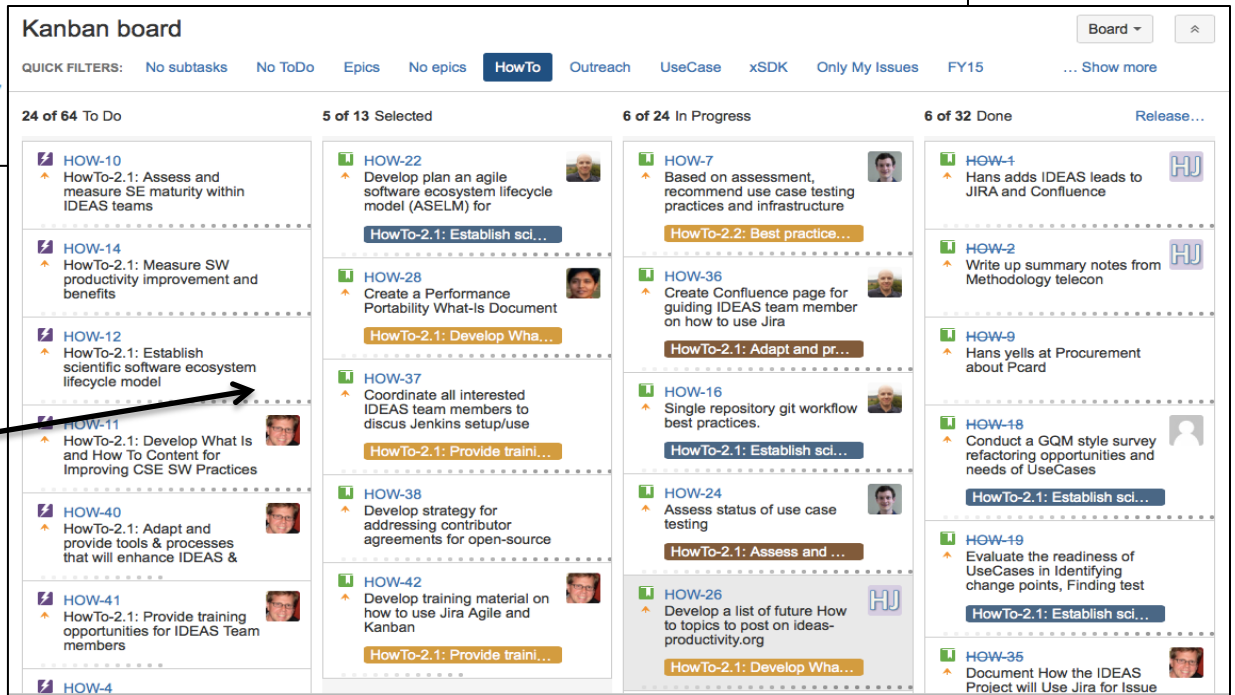
Managing IDEAS Project Activities Using JIRA Agile and Kanban

Created by Michael Heroux, last modified by Jim Willenbring about an hour ago

Contents:

Developer Guide, on Confluence site

Kanban Board, on Jira site.
Four columns:
- To Do
- Selected
- In Progress
- Done



Sandia National Laboratories

Message to This Audience

Improve your issue tracking habits:

- *Nothing -> Desktop/todo.txt*
- *Desktop/todo.txt -> clone/todo.txt*
- *clone/todo.txt -> GitHub Issues*
- *GitHub Issues -> GitHub Issues + Kanban*
  or Jira + Kanban

Three Application Design Strategies for Productivity & Sustainability

Strategy 1: Array and Execution Abstraction

Performance Portable Threaded Computations using Kokkos

Primary Developers: H.C. Edwards, C. Trott

Three Parallel Computing Design Points

- Terascale Laptop: Uninode-Multicore/Manycore/GPU
- Petascale Deskside: Multinode-Multicore/Manycore/GPU
- Exascale Center: Manynode-Multicore/Manycore/GPU

Goal: Make
Petascale = Terascale + more
Exascale = Petascale + more

Common Element

Most applications will not adopt an exascale programming strategy that is incompatible with tera and peta scale.

Performance Portability Challenge:

Best (decent) performance requires computations to implement architecture-specific memory access patterns

- CPUs (and Xeon Phi)
  - Core-data affinity: consistent NUMA access (first touch)
  - Array alignment for cache-lines and vector units
  - Hyperthreads' cooperative use of L1 cache
- GPUs
  - Thread-data affinity: coalesced access with cache-line alignment
  - Temporal locality and special hardware (texture cache)
- Array of Structures (AoS) vs. Structure of Arrays (SoA) dilemma
  - i.e., architecture specific data structure layout and access

➢ This has been the *wrong* concern

The right concern: Abstractions for Performance Portability?

Kokkos: Execution and memory space abstractions

- What is Kokkos:
  - C++ (C++11) template meta-programming library, part of (and not) Trilinos.
  - Compile-time polymorphic multi-dimensional array classes.
  - Parallel execution patterns: For, Reduce, Scan.
  - Loop body code: Functors, lambdas.
  - Tasks: Asynchronous launch, Futures.
- Available independently (outside of Trilinos):
  - https://github.com/kokkos/
- Getting started:
  - GTC 2015 Content:
    - http://on-demand.gputechconf.com/gtc/2015/video/S5166.html
    - http://on-demand.gputechconf.com/gtc/2015/presentation/S5166-H-Carter-Edwards.pdf
  - Programming guide doc/Kokkos_PG.pdf.

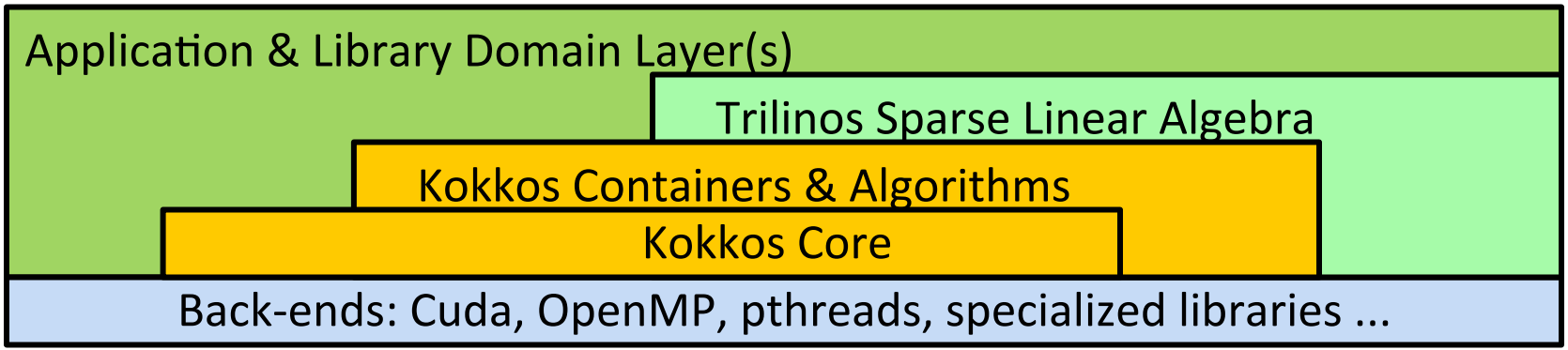
Kokkos' Performance Portability Answer

Integrated mapping of thread parallel computations and multidimensional array data onto manycore architecture

1. Map user's parallel computations to threads
   - Parallel pattern: foreach, reduce, scan, task-dag, ...
   - Parallel loop/task body: C++11 lambda or C++98 functor
2. Map user's datum to memory
   - Multidimensional array of datum, *with a twist*
   - Layout : multi-index (i,j,k,...) memory location
   - Kokkos *chooses* layout for architecture-specific memory access pattern
   - Polymorphic multidimensional array
3. Access user datum through special hardware (bonus)
   - GPU texture cache to speed up read-only random access patterns
   - Atomic operations for thread safety

Layered Collection of C++ Libraries

- Standard C++, Not a language extension
  - *Not* a language extension: OpenMP, OpenACC, OpenCL, CUDA
  - In *spirit* of Intel's TBB, NVIDIA's Thrust & CUSP, MS C++AMP, ...
- Uses C++ template meta-programming
  - Previously relied upon C++1998 standard
  - Now require C++2011 for lambda functionality
    - Supported by Cuda 7.0; full functionality in Cuda 7.5
  - Participating in ISO/C++ standard committee for core capabilities



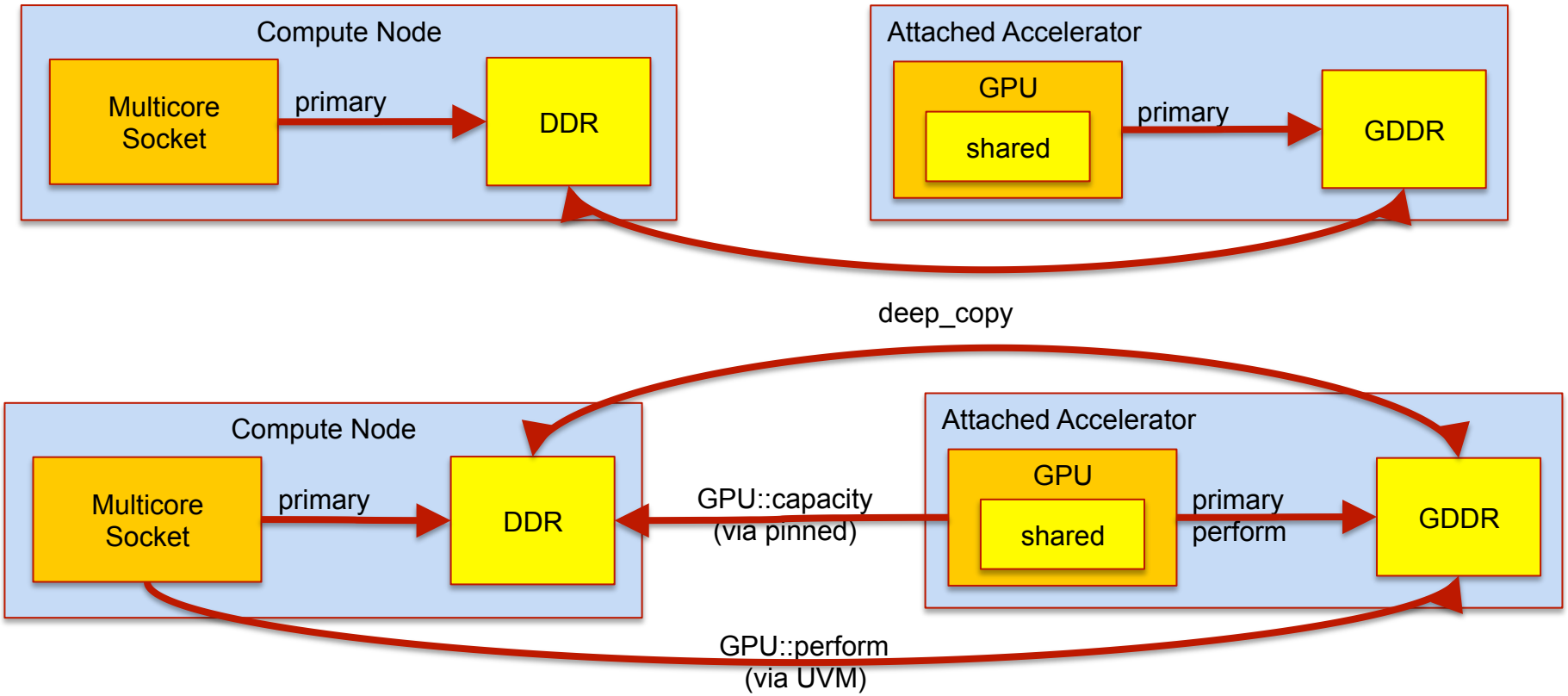
Abstractions: Spaces, Policies, and Patterns

- Memory Space : where data resides
  - Differentiated by performance; e.g., size, latency, bandwidth
- Execution Space : where functions execute
  - Encapsulates hardware resources; e.g., cores, GPU, vector units, ...
  - Denote accessible memory spaces
- Execution Policy : how (and where) a user function is executed
  - E.g., data parallel range : concurrently call function(i) for i = [0..N)
  - User's function is a C++ functor or C++11 lambda
- Pattern: parallel_for, parallel_reduce, parallel_scan, task-dag, ...
- Compose: pattern + execution policy + user function; e.g.,

```
parallel_pattern( Policy<Space>, Function);
```

  - Execute *Function* in *Space* according to *pattern* and *Policy*
- Extensible spaces, policies, and patterns

Examples of Execution and Memory Spaces

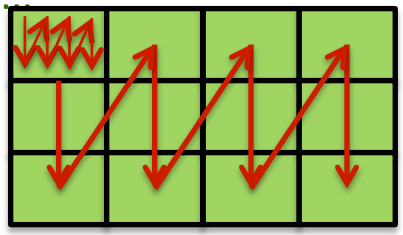


Polymorphic Multidimensional Array View

- View< double**[3][8] , Space > a("a",N,M);
  - Allocate array data in memory Space with dimensions [N][M][3][8]
  - ? C++17 improvement to allow View<double[ ][ ][3][8],Space>
- a(i,j,k,l) : User's access to array datum
  - "Space" accessibility enforced; e.g., GPU code cannot access CPU memory
  - Optional array bounds checking of indices for debugging
- View Semantics: View<double**[3][8],Space> b = a ;
  - A shallow copy: 'a' and 'b' are *pointers* to the same allocated array data
  - Analogous to C++11 std::shared_ptr
- deep_copy(destination_view , source_view);
  - Copy data from 'source_view' to 'destination_view'
  - ➢ Kokkos policy: never hide an expensive deep copy operation

Polymorphic Multidimensional Array Layout

- Layout mapping : a(i,j,k,l) → memory location
  - Layout is polymorphic, defined at compile time
  - Kokkos chooses default array layout appropriate for "Space"
  - E.g., row-major, column-major, Morton ordering, dimension padding, ...
- User can specify Layout : View< ArrayType, *Layout*, Space >
  - Override Kokkos' default choice for layout
  - Why? For compatibility with legacy code, algorithmic performance tuning, ...
- Example Tiling Layout
  - View<double**,Tile<8,8>,Space> m("matrix",N,N);
  - Tiling layout transparent to user code : m(i,j) unchanged
  - Layout-aware algorithm extracts tile subview



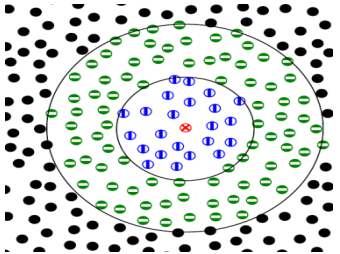
Performance Impact of Data Layout

Sandia National Laboratories

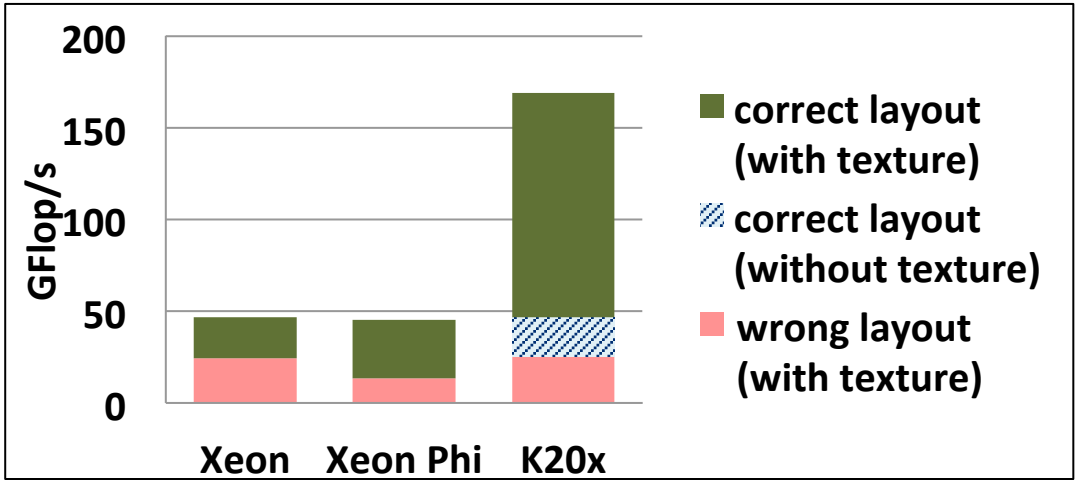
- Molecular dynamics computational kernel in miniMD
- Simple Lennard Jones force model:
- Atom neighbor list to avoid N^2 computations

$$F_i = \sum_{j,\, r_{ij} < r_{cut}} 6\varepsilon \left[\left(\frac{\varsigma}{r_{ij}}\right)^{7} - 2\left(\frac{\varsigma}{r_{ij}}\right)^{13} \right]$$

```
pos_i = pos(i);
for( jj = 0; jj < num_neighbors(i); jj++) {
  j = neighbors(i,jj);
  r_ij = pos(i,0..2) - pos(j,0..2); //random read 3 floats
  if (|r_ij| < r_cut) f_i += 6*e*((s/r_ij)^7 - 2*(s/r_ij)^13)
}
f(i) = f_i;
```



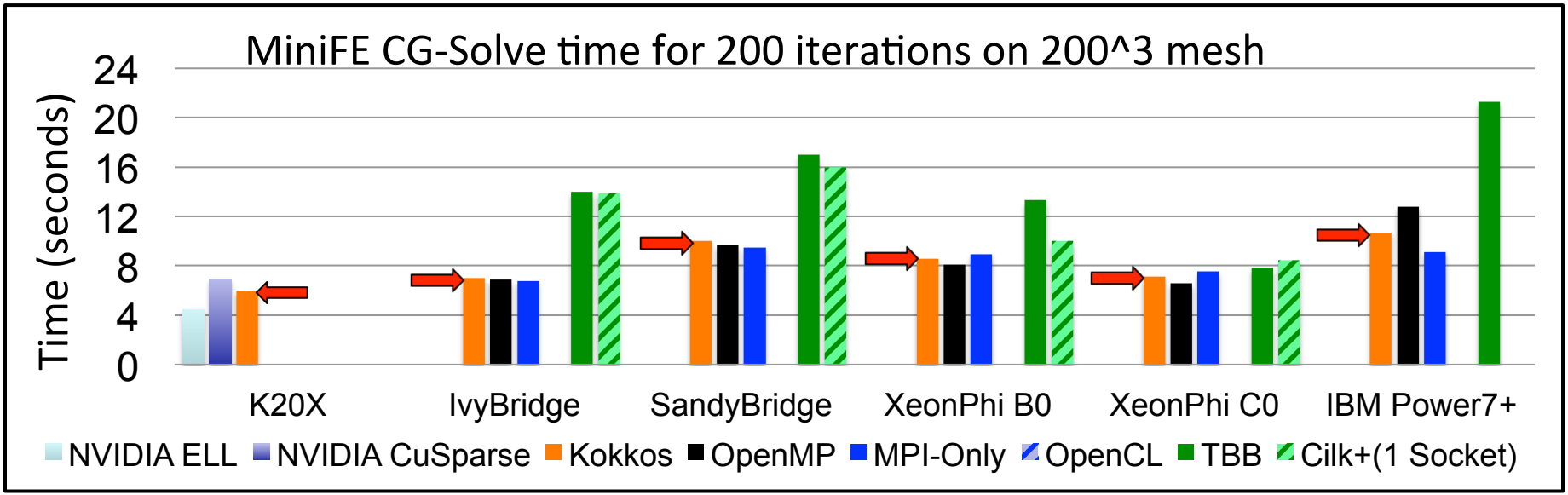
- Test Problem
  - 864k atoms, ~77 neighbors
  - 2D neighbor array
  - Different layouts CPU vs GPU
  - Random read 'pos' through GPU texture cache
- Large performance loss with wrong array layout



Performance Overhead?

Kokkos is competitive with native programming models

- Regularly performance-test mini-applications on Sandia's ASC/CSSE test beds
- MiniFE: finite element linear system iterative solver mini-app
  - Compare to versions with architecture-specialized programming models



Kokkos Summary

Sandia National Laboratories

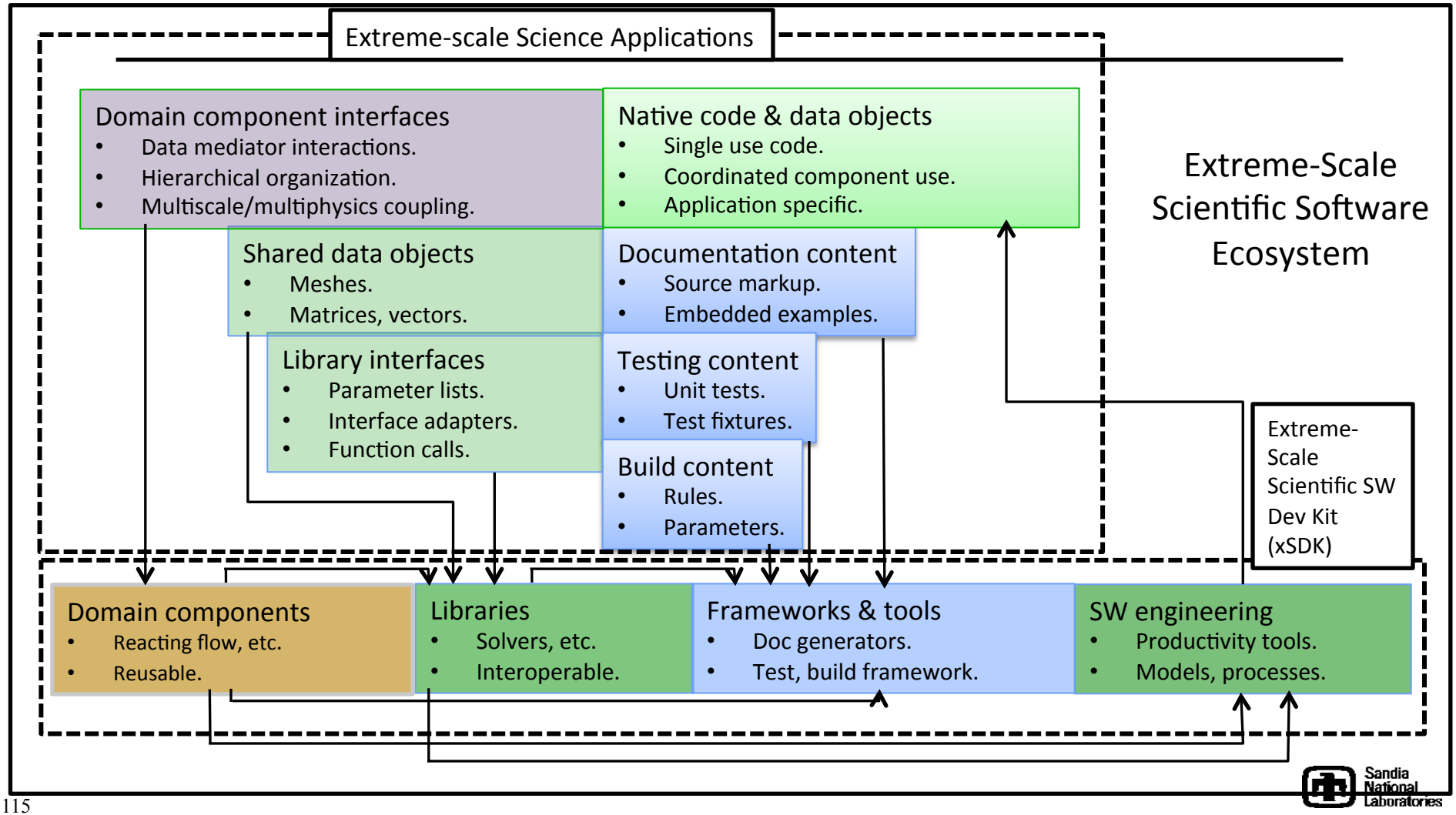
- Technical: Performance Portability for C++ Applications
  - Production library on Multicore CPU, Intel Xeon Phi (KNC), IBM Power 8, and NVIDIA GPU (Kepler); *AMD Fusion in progress*
    - Tutorial at Sandia Sept 1-2, 2015; tutorial at Supercomputing'15
  - Integrated mapping of applications' computations *and* data
    - Other programming models fail to map data and limit performance portability
  - Future proofing with designed-in extensibility
- Programmatic: History and Collaborations led to success
  - Evolving strategic vision, concepts, requirements, and design
  - Persistent advocacy and investment by DOE programs and Sandia LDRDs
  - Numerous strategic collaborations across ASC program elements, labs, universities, and vendors
- Goal: ISO/C++ 2020 Standard supplants Kokkos functionality
  - Strategic, persistent effort requiring multi-lab membership & advocacy

Message to This Audience

Consider an array/patterns library, e.g., Kokkos.

Develop your own light-weight array abstraction layer.

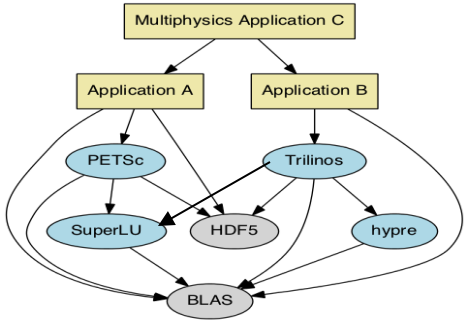
Strategy 2: Application Composition and Software Eco-systems



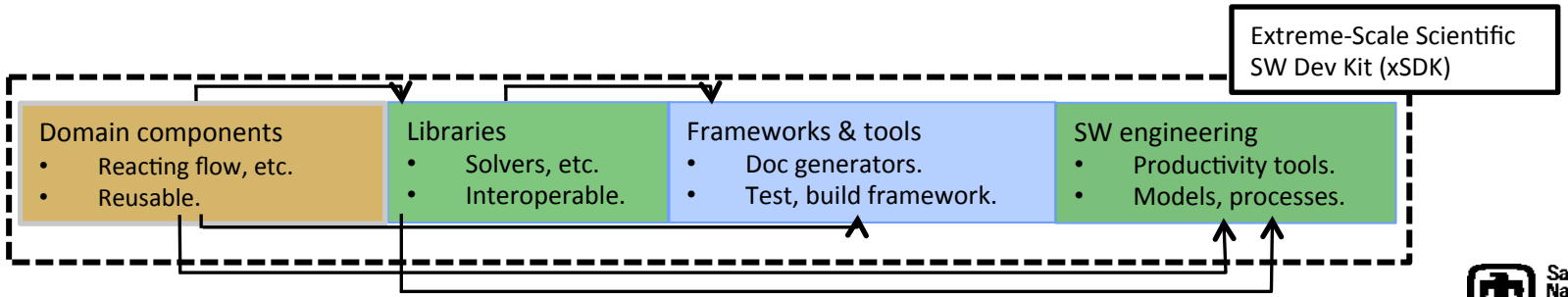
xSDK Foundations

Focus: Increasing the functionality, quality, and interoperability of important scientific libraries and development tools

- xSDK foundations: Seeking community feedback: https://ideas-productivity.org/resources/xsdk-docs/
  - xSDK package compliance standards
  - xSDK standard configure and CMake options
- Library interoperability
- Designing for performance portability

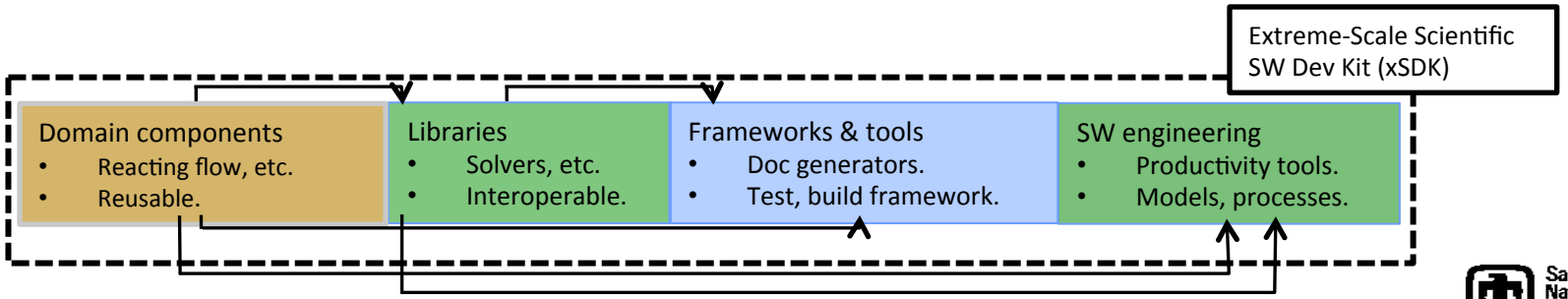


Standard xSDK package installation interface facilitates combined use of xSDK libraries (initially hypre, PETSc, SuperLU, Trilinos), as needed by BER use cases and other multiphysics apps.



xSDK focus

- Common configure and link capabilities
  - xSDK users need full and consistent access to all xSDK capabilities
  - Namespace and version conflicts make simultaneous build/link of xSDK difficult
  - Determining an approach that can be adopted by any library or components development team for standardized configure/link processes
- Library interoperability
- Designing for performance portability



Standard xSDK package installation interface

Motivation: Obtaining, configuring, and installing multiple independent software packages is tedious and error prone.

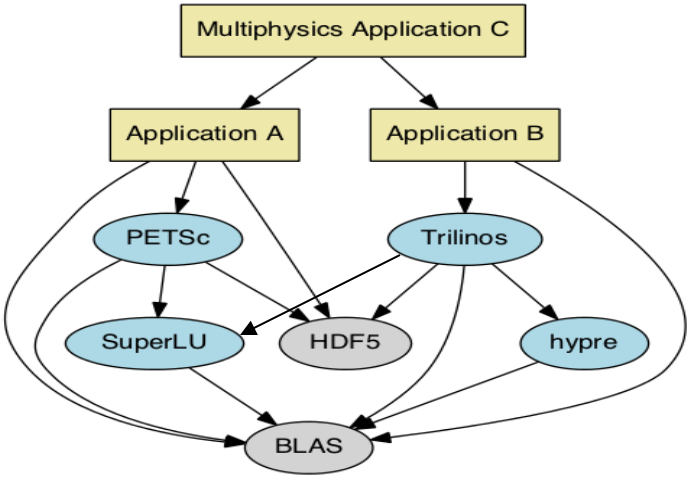
- Need consistency of compiler (+version, options), 3rd-party packages, etc.

Approach: Define a *standard xSDK package installation interface* to which all xSDK packages will subscribe and be tested

Accomplishments:

- Work on implementations of the standard by the hypre, PETSc, SuperLU, and Trilinos developers
- PETSc can now use the “scriptable” feature of the installers to simultaneously install hypre, PETSc, SuperLU, Trilinos with consistent compilers and ‘helper’ libraries.

xSDK Build Example



Impact: Foundational step toward seamless combined use of xSDK libraries, as needed by BER use cases and other multiphysics apps

xSDK Minimum Compliance Requirements:

- M1. Each xSDK compliant package must support the the standard xSDK cmake/ configure options.
- M2. Each xSDK package must provide a comprehensive test suite that can be run by users and does not require the purchase of commercial software
- M3. Each xSDK compliant package that utilizes MPI must restrict its MPI operations to MPI communicators that are provided to it and not use directly MPI_COMM_WORLD.
- M4. Each package team must do a 'best effort' at portability to key architectures, including standard Linux distributions, GNU, Clang, vendor compilers, and target machines at ALCF, NERSC, OLCF. Apple Mac OS and Microsoft Windows support are recommended.
- M5. Each package team must provide a documented, reliable way to contact the development team; this may be by email or a website. The package teams should not require users to join a generic mailing list (and hence receive irrelevant email they must wade through) in order to report bugs or request assistance.
- M6 – 11…

https://ideas-productivity.org/resources/xsdk-docs

xSDK Recommended Compliance Requirements:

- R1. It is recommended that each package have a public repository, for example at github or bitbucket, where the development version of the package is available. Support for taking pull requests is also recommended.
- R2. It is recommend that all libraries be tested with valgrind for memory corruption issues while the test suite is run.
- R3. It is recommended that each package adopt and document a consistent system for propagating/returning error conditions/exceptions and provide an API for changing the behavior.
- R4. It is recommended that each package free all system resources it has acquired as soon as they are no longer needed.

Typical Trilinos Cmake Script (edison)

```
cmake \
-D MPI_CXX_COMPILER="CC" \
-D MPI_C_COMPILER="cc" \
-D MPI_Fortran_COMPILER="ftn" \
-D Teuchos_ENABLE_STACKTRACE:BOOL=OFF \
-D Teuchos_ENABLE_LONG_LONG_INT:BOOL=ON \
-D Trilinos_ENABLE_Tpetra:BOOL=ON \
-D Tpetra_ENABLE_TESTS:BOOL=ON \
-D Tpetra_ENABLE_EXAMPLES:BOOL=ON \
-D Tpetra_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON \
-D Teuchos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON \
-D TPL_ENABLE_MPI:BOOL=ON \
-D CMAKE_INSTALL_PREFIX:PATH="$HOME/opt/Trilinos/tpetraEval" \
-D BLAS_LIBRARY_DIRS="$LIBSCI_BASE_DIR/gnu/lib" \
-D BLAS_LIBRARY_NAMES="sci" \
-D LAPACK_LIBRARY_DIRS="$LIBSCI_BASE_DIR/gnu/lib" \
-D LAPACK_LIBRARY_NAMES="sci" \
-D CMAKE_CXX_FLAGS="-O3 -ffast-math -funroll-loops" \
\
..
```

WebTrilinos

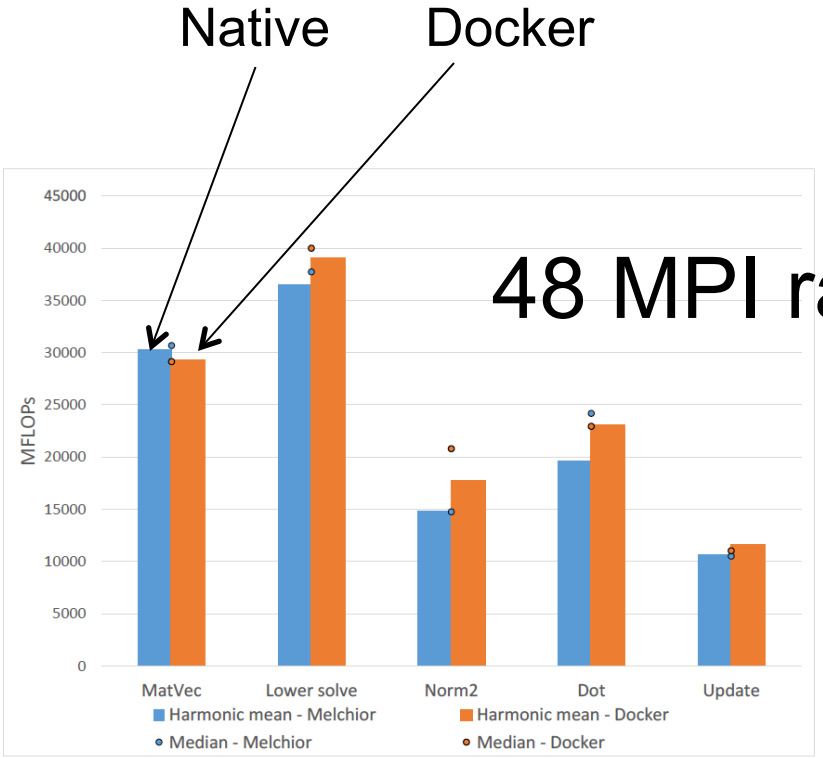
Trilinos usage via Docker

- WebTrilinos Tutorial
  - https://hub.docker.com/r/sjdeal/webtrilinos
- http://johntfoster.github.io/posts/peridigm-without-building-via-Docker.html
  - docker pull johntfoster/trilinos
  - docker pull johntfoster/peridigm
  - docker run --name peridigm0 -d -v `pwd`:/output johntfoster/peridigm \
    Peridigm fragmenting_cylinder.peridigm
  - Etc…

First Docker MPI Results (Sean Deal)

SJU Cluster, Epetra Basic Perf Test

- MatVec
- Lower Solve
- Norm2, Dot, Update
- Harmonic mean of 5 tests
- 4M Eq per proc

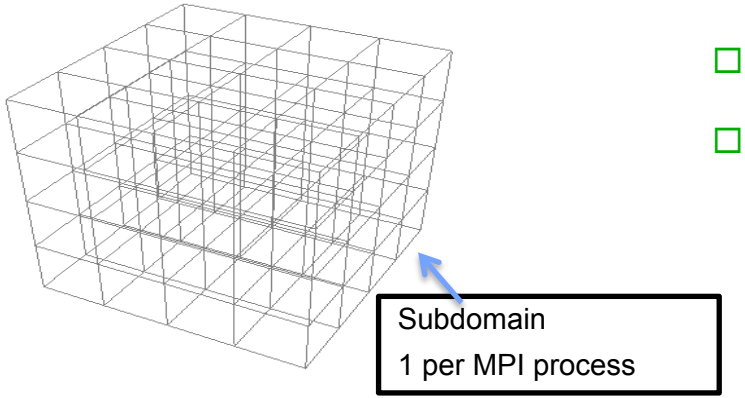


Message to This Audience

Consider what software ecosystem(s) you want your software to be part of and use.

Strategy 3: Toward a New Application Architecture

Classic HPC Application Architecture

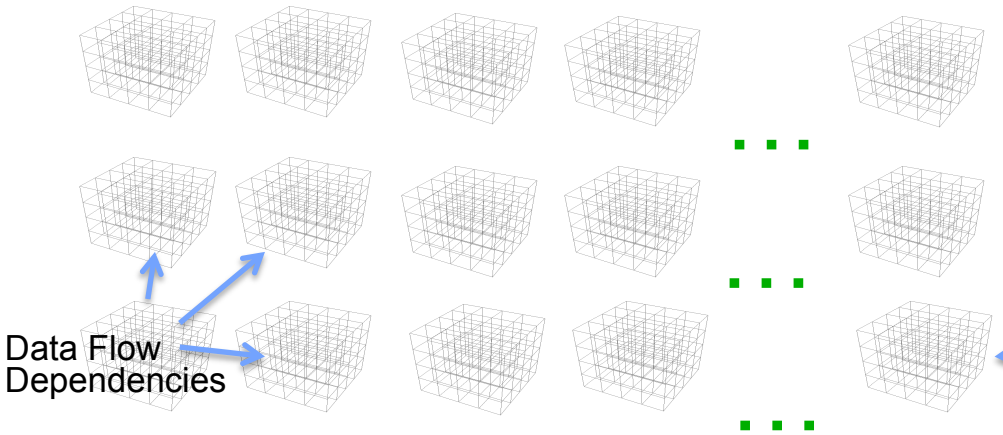


- Logically Bulk-Synchronous, SPMD
- Basic Attributes:
  - Halo exchange.
  - Local compute.
  - Global collective.

- Strengths:
  - Portable to many specific system architectures.
  - Separation of parallel model (SPMD) from implementation (e.g., message passing).
  - Domain scientists write sequential code within a parallel SPMD framework.
  - Supports traditional languages (Fortran, C).

- Weaknesses:
  - Not well suited (as-is) to emerging manycore systems.
  - Unable to exploit functional on-chip parallelism.
  - Difficult to tolerate dynamic latencies.
  - Difficult to support task/compute heterogeneity.

Task-centric/Dataflow Application Architecture



- Patch: Logically connected portion of global data. Ex: subdomain, subgraph.
- Task: Functionality defined on a patch.
- Many tasks on many patches.

- Strengths:
  - Portable to many specific system architectures.
  - Separation of parallel model from implementation.
  - Domain scientists write sequential code within a parallel framework.
  - Supports traditional languages (Fortran, C).
  - Similar to SPMD in many ways.

- More strengths:
  - Well suited to emerging manycore systems.
  - Can exploit functional on-chip parallelism.
  - Can tolerate dynamic latencies.
  - Can support task/compute heterogeneity.

Recap: Trilinos

Trilinos provides a rich collection of linear solvers:

- Uniform access to many direct sparse solvers.
- An extensive collection of iterative methods:
  - Classic single RHS: CG, GMRES, etc.
  - Pseudo-block: Multiple independent systems.
  - Recycling: Multiple sequential RHS.
  - Block: Multiple simultaneous RHS.
- A broad set of preconditioners:
  - Domain decomposition.
  - Algebraic smoothers.
  - AMG.
- Composable, extensible framework.
  - RowMatrix and Operator base classes enable user-define operators.
  - Multi-physic and multi-scale operators composed from Trilinos parts.
- Template features enable:
  - Variable precision, complex values.
- Significant R&D in:
  - Thread-scalable algorithms, kernels.
  - Resilient methods.

Recap: Future Ecosystem Efforts.

- Thread-scalable algorithms making steady progress: “easy”.
- Resilience strategies too, and reliability will persist until we are ready: “easy”.
- Big task: Transforming application base to new systems and beyond.
- SW engineering focus is important for HPC:
  - Pursuing efficiency negatively impacts many other quality metrics.
- Productive application designs will require disruptive changes:
  - Array and execution abstractions needed for portability.
  - Reuse via composition is attractive (think Android/iOS, Docker environments).
  - A Task-centric/dataflow app architecture is very attractive for performance portability.

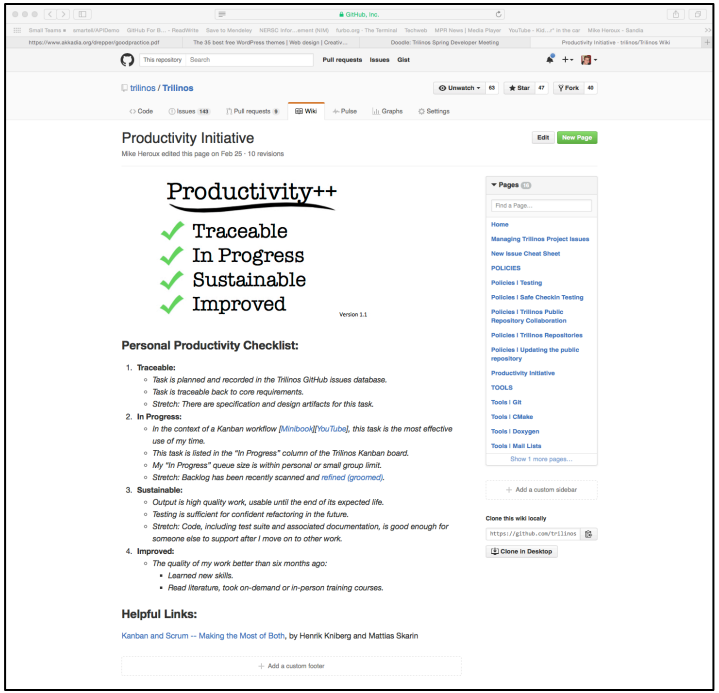
Final Thought: Commitment to Quality

Canadian engineers' oath (taken from Rudyard Kipling):

My Time I will not refuse;
my Thought I will not grudge;
my Care I will not deny
toward the honour, use,
stability and perfection of
any works to which I may be
called to set my hand.

http://commons.bcit.ca/update/2010/11/bcit-engineering-graduates-earn-their-iron-rings

Productivity++ Initiative



https://github.com/trilinos/Trilinos/wiki/Productivity---Initiative

Sandia National Laboratories

Quiz (True/False)

1. Building an app via reusable SW components is always a good idea. **False**
2. Use of third party solvers is always a good idea. **True**
3. Control inversion is a means of customizing SW ecosystem behavior. **True**
4. Framework use must be an "all-in" commitment. **False**
5. Performance portability requires data structure abstractions. **True (for now)**
6. Operator abstractions enable sophisticated solution algorithms. **True**
7. Algorithm development for massive concurrency is "easy" part of our job. **True**
8. Code optimization is always a good idea for HPC software. **False**
9. Containerization capabilities are important for scientific SW. **True**
10. Productivity must be a first-order, explicit focus for future scientific SW. **True**

To Learn More

- Visit the Trilinos Tutorial Site:
  - https://github.com/trilinos/Trilinos_tutorial/wiki/
- Visit the IDEAS Scientific SW Productivity Site:
  - https://ideas-productivity.org