Recent Advancements and Future Plans for Next-Generation Sparse Solvers in Trilinos

SPPEXA Symposium 2016
January 25th, 2016

Heidi K. Thornquist
Sandia National Laboratories
with contributions from the Trilinos Development Team

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.

Sandia National Laboratories

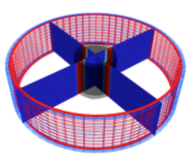
- R&D 100 Winner
- Open Source
  - Accessible via GitHub

Laptops to Leadership systems

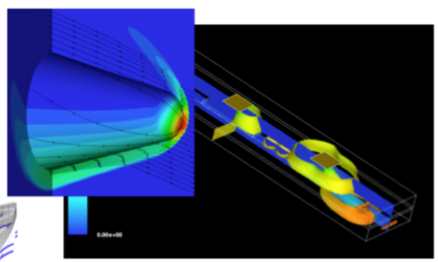
Optimal Kernels to Optimal Solutions:

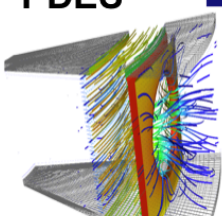
- Geometry, Meshing
- Discretizations, Load Balancing.
- Scalable Linear, Nonlinear, Eigen, Transient, Optimization, UQ solvers.
- Scalable I/O, GPU, Manycore

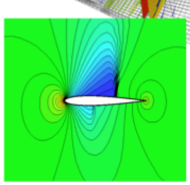
- 60 Packages.
- Binary distributions:
  - Cray LIBSCI
  - Debian, Ubuntu
  - Intel (in process)

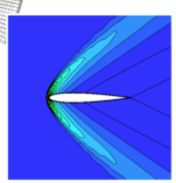


PDES



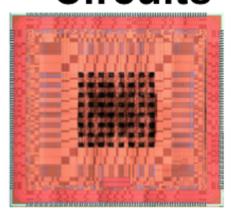




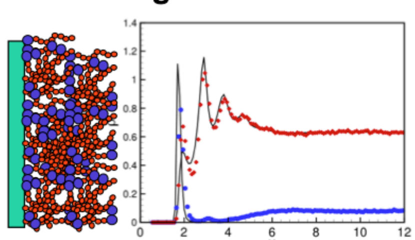


And More…

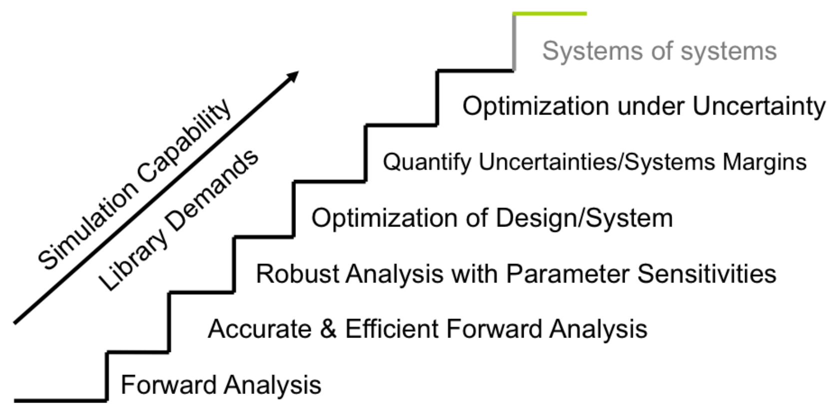
Circuits



Inhomogeneous Fluids



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.

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

Sandia National Laboratories

Compile-time Polymorphism

Software delivery:

- Essential Activity

How can we:

- Implement mixed precision algorithms?
- Implement generic fine-grain parallelism?
- Support hybrid CPU/GPU computations?
- Support extended precision?
- Explore redundant computations?
- Prepare for both exascale “swim lanes”?

C++ templates only sane way:

- Moving to completely templated Trilinos libraries.
- Other important benefits.
- **A usable stack exists now in Trilinos**.

Template Benefits:

- Compile time polymorphism.
- True generic programming.
- No runtime performance hit.
- Strong typing for mixed precision.
- Support for extended precision.
- Many more…

Template Drawbacks:

- Huge compile-time performance hit:
  - But good use of multicore :)
  - Eliminated for common data types.
- Complex notation:
  - Esp. for Fortran & C programmers).
  - Can insulate to some extent.

Solver Software Stack

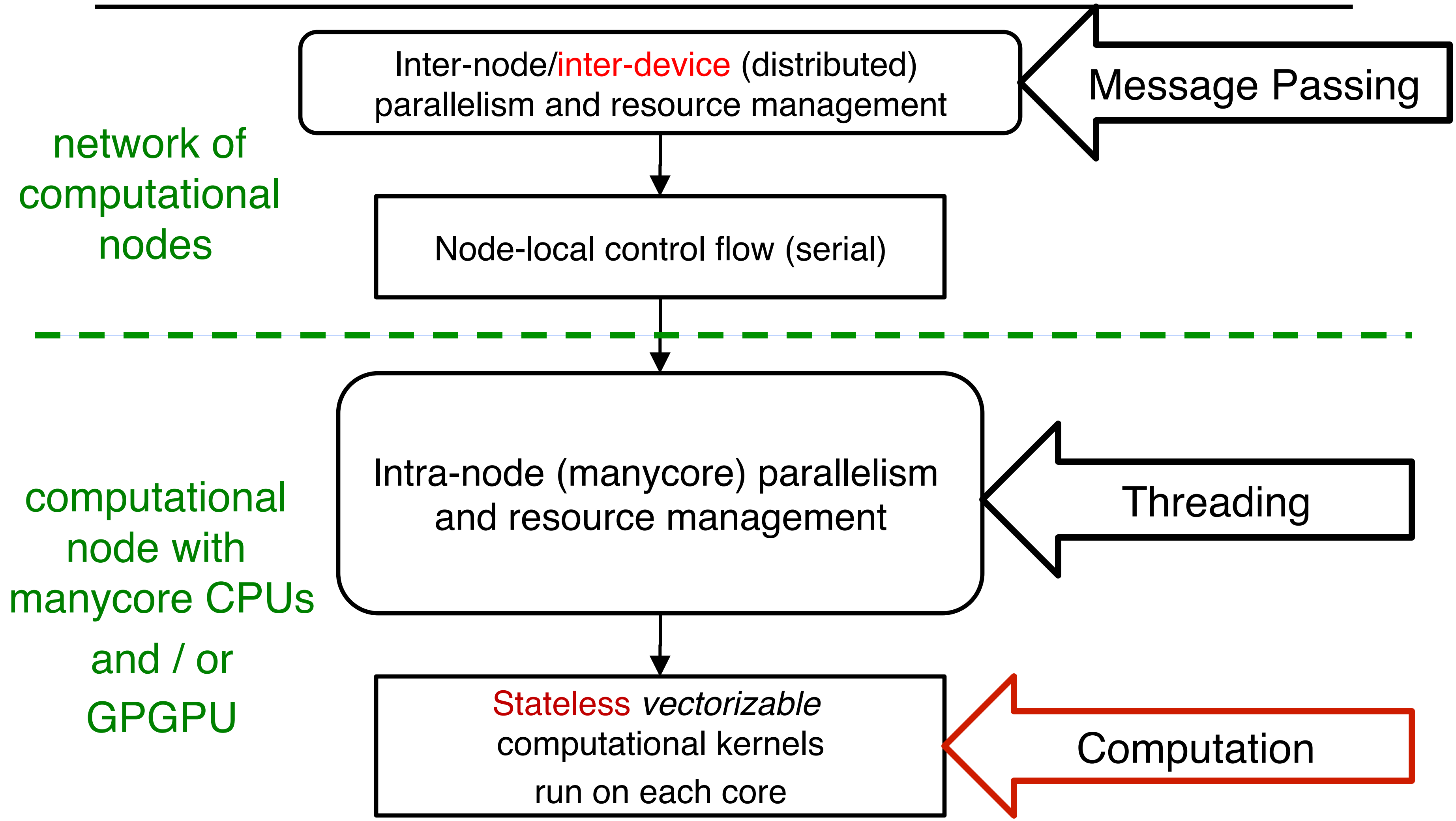
Trilinos

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

Problem	Formulation	Packages
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 $F(x) = 0 \quad x \in \Re^n$	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) $\nu \in \Re^n, \quad \lambda \in$	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)

Parallel Programming Model: Multi-level/Multi-device



Sandia National Laboratories

Tpetra and Kokkos Packages

- Tpetra is a distributed linear algebra library.
  - Similar to Trilinos/Epetra:
    - Provides maps, vectors, sparse matrices and abstract linear operators
  - Heavily exploits templated C++
  - Employs hybrid (distributed + shared) parallelism via Kokkos
- Kokkos is an API for shared-memory parallel nodes.
  - Provides parallel_for and parallel_reduce skeletons
  - Provides local, shared-memory parallel linear algebra
  - Currently supports multiple shared-memory APIs:
    - OpenMP
    - Pthreads
    - NVIDIA CUDA-capable GPUs

Belos: Iterative linear solvers

- Provide a generic framework for developing iterative algorithms for solving large-scale linear systems.
- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - Operator-vector products: `Belos::MultiVecTraits`, `Belos::OperatorTraits`
  - Orthogonalization: `Belos::OrthoManager`, `Belos::MatOrthoManager`
  - Status tests: `Belos::StatusTest`, `Belos::StatusTestResNorm`
  - Iteration kernels: `Belos::Iteration`
  - Solver managers: `Belos::SolverManager`
  - Linear problem: `Belos::LinearProblem`
- Currently has solver managers for several linear solvers:
  - GMRES: Single-vector, block, pseudo-block, flexible
  - CG: Single-vector, block, pseudo-block
  - TFQMR, BiCGStab, MINRES
  - Least squares: LSQR
  - Recycling solvers: GCRO-DR, RCG
  - Seed solvers: PCPG, GMRES poly
- Can solve:
  - Hermitian, non-Hermitian linear problems
  - Real, complex-valued, arbitrary linear problems
    - arbitrary precision can be limited by LAPACK

Sandia National Laboratories

Belos Arbitrary Precision Example

- Using Tpetra for underlying linear algebra just requires different template arguments in Belos
- Tpetra objects are templated on the underlying data types:

```
MultiVector<Scalar, LO, GO, Node> …
Operator<Scalar, LO, GO, Node> …
```

  – `LO=GO=int, Node=Serial`

Scalar	float	double	double-double	quad-double
Solve time (s)	2.6	5.3	29.9	76.5
Accuracy	10^{-6}	10^{-12}	10^{-24}	10^{-48}

Speedup of float over double in Belos linear solver.

float	double	speedup
18 s	26 s	1.42x

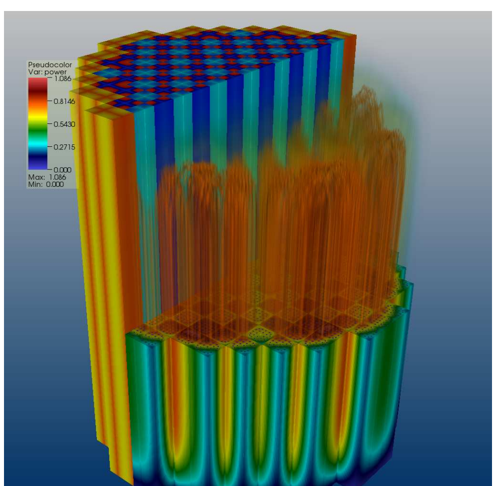
Anasazi: Iterative eigensolvers

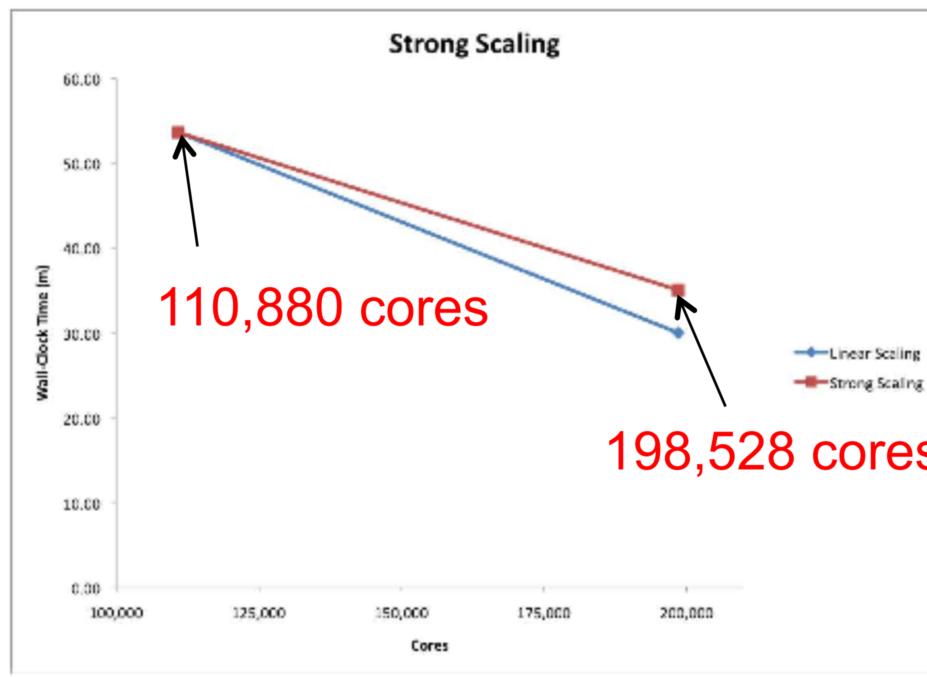
- Provide a generic framework for developing iterative algorithms for solving large-scale eigenproblems.
- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - Operator-vector products: `Anasazi::MultiVecTraits`, `Anasazi::OperatorTraits`
  - Orthogonalization: `Anasazi::OrthoManager`, `Anasazi::MatOrthoManager`
  - Status tests: `Anasazi::StatusTest`, `Anasazi::StatusTestResNorm`
  - Iteration kernels: `Anasazi::Eigensolver`
  - Eigensolver managers: `Anasazi::SolverManager`
  - Eigenproblem: `Anasazi::Eigenproblem`
  - Sort managers: `Anasazi::SortManager`
- Currently has solver managers for several eigensolvers:
  - Block Krylov-Schur
  - Block Davidson
  - LOBPCG
  - Generalized Davidson
  - TraceMin
- Can solve:
  - Standard and generalized eigenproblems
  - Hermitian and non-Hermitian eigenproblems
  - Real or complex-valued eigenproblems {arbitrary precision limited by LAPACK}

Sandia National Laboratories

Anasazi & Denovo Example

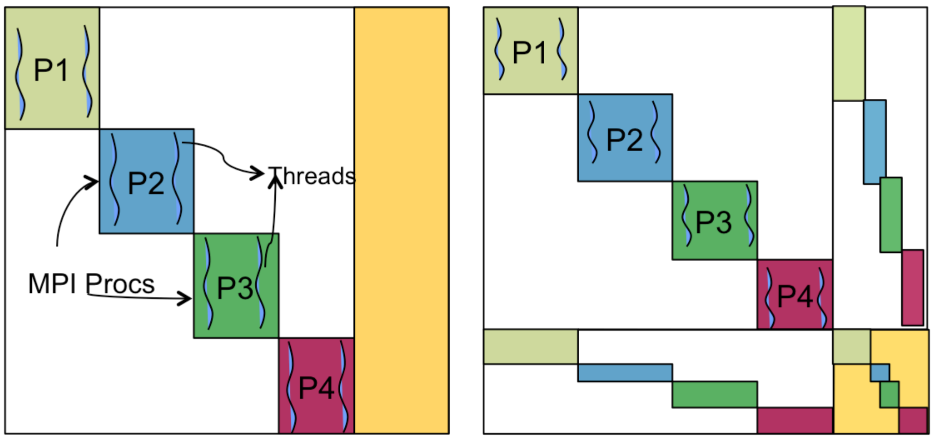
- Denovo is a 3D, discrete ordinates multigroup radiation transport code for radiation shielding and reactor physics applications at ORNL.
- Block Krylov-Schur eigensolver used to solve the k-eigenvalue problem.
- Example: Generic Westinghouse PWR-900 nuclear reactor core
  - 2 energy groups and a 578 x 578 x 700 mesh
    - ~234 million cells
    - 78 billion unknowns
  - Computations performed on Jaguar, Cray XT-5 @ ORNL.





Preconditioners

- Convergence of iterative linear solvers and eigensolvers can be accelerated by the use of preconditioners.
- Templated preconditioner packages: Ifpack2, MueLu, ShyLU



Hypergraph/Graph based ordering of the matrix for the ShyLU

- Subdomain solvers or smoothers have to adapt to hierarchical architectures.
  - One MPI process per core cannot exploit intra-node parallelism.
  - One subdomain per MPI process hard to scale. (due to increase in the number of iterations)
- ShyLU (Scalable Hybrid LU) is hybrid
  - In the mathematical sense (direct + iterative) for robustness.
  - In the parallel programming sense (MPI + Threads) for scalability.
- More robust than simple preconditioners and scalable than direct solvers.
- ShyLU is a subdomain solver where a subdomain is not limited to one MPI process.

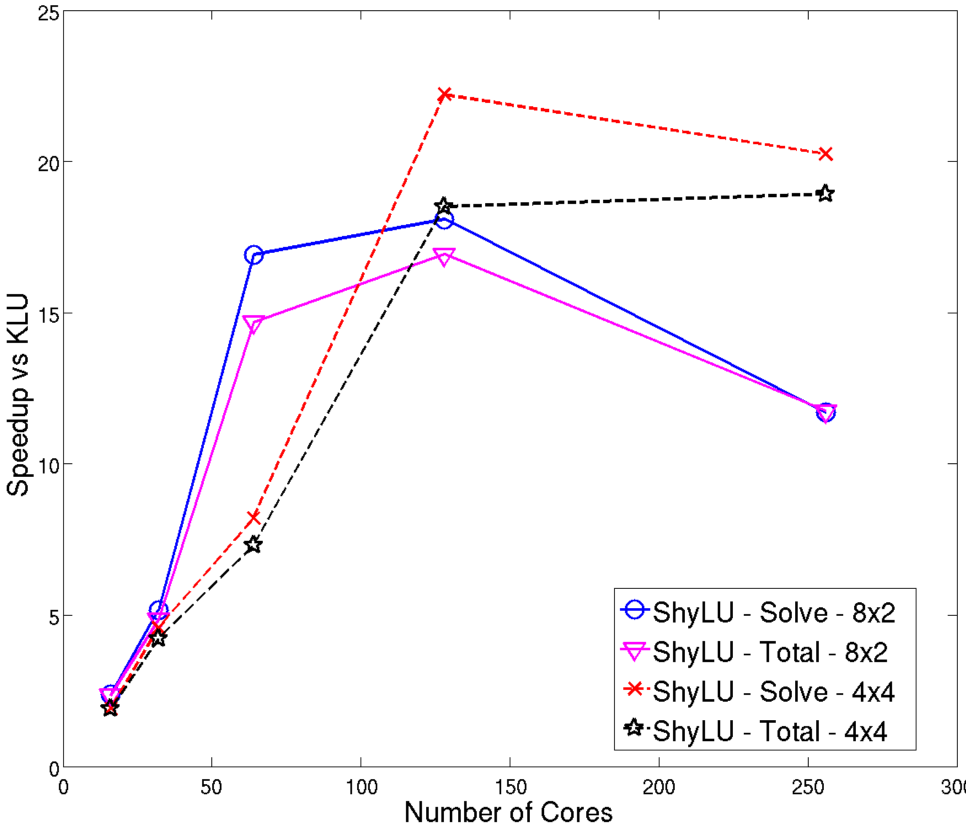
Sandia National Laboratories

ShyLU & Xyce Example

- This solution approach was necessary for efficient simulation of new Sandia-designed ASICs

$$f(x(t)) + \frac{dq(x(t))}{dt} = b(t)$$

- Ill conditioned, heterogeneous linear system structure
- Example:
  - 1645693 total devices, N = 1944792
  - Single KLU solve: ~ 40 sec.
  - Single SuperLU solve: ~ 200 sec.
  - ShyLU: 4 MPI procs -> rows(S) = 1854



Sandia National Laboratories

Node-level Solvers

- Preconditioners require efficient **node-level** solvers that utilize thread-level parallelism.
  - Coarse-level solve in multigrid
  - Subdomain solve in domain decomposition preconditioner

- Ongoing efforts in developing node-level solvers include:
  - Task-parallel incomplete Cholesky factorizations (**Tacho**, Kyungjoo Kim)
  - Gauss-Seidel with coloring (Mehmet Deveci)
  - Multi-threaded sparse triangular solve (Andrew Bradley)
  - New sparse-direct solver framework that exploits both the multiple levels in matrix structure and memory hierarchy (**Basker**, Joshua Booth)
    - First parallel implementation of Gilbert-Peierls algorithm
    - Enables efficient block solution methods for frequency-domain analysis, PCE, etc.

Sandia National Laboratories

Complimentary Efforts

- Tpetra / Kokkos “setup time” improvements
  - Interfaces for thread-parallel graph and matrix fill
  - Node-level kernels for matrix-matrix multiply and aggregation
- Fault tolerant iterative methods
  - Bit flips (James Elliott)
  - Process failures (Keita Teranishi)
- Communication-avoiding solvers
  - Integrate communication avoiding solvers into Trilinos
  - Develop communication avoiding preconditioners to accelerate iterative solver convergence (S. Rajamanickam, et al)

Trilinos Availability / Information

- **Trilinos and related packages are available via LGPL or BSD.**
- **Current release (12.4), unlimited availability.**
- **Trilinos Awards:**
  - **2004 R&D 100 Award.**
  - **SC2004 HPC Software Challenge Award.**
  - **Sandia Team Employee Recognition Award.**
  - **Lockheed-Martin Nova Award Nominee.**
- **More information:**
  - **http://trilinos.org**
  - **https://github.com/trilinos/trilinos**
- **Annual Forums:**
  - **Annual Trilinos User Group (TUG) Meeting in November @ SNL**
    - **talks and video available for download**
  - **Spring Developer Meeting, May @ SNL**
  - **EuroTUG 2016, April 18-20, LRZ, Garching, Germany**

Sandia National Laboratories