The Zoltan Toolkit

Karen Devine

Scalable Algorithms Department
Sandia National Laboratories

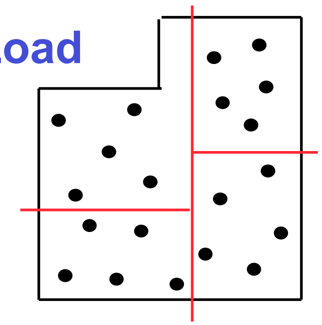
FASTMath SciDAC Institute

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.

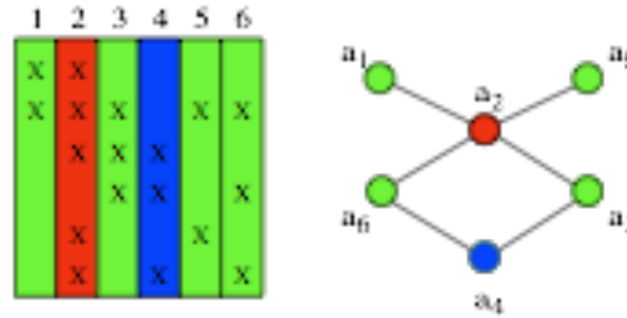
The Zoltan Toolkit

Library of parallel combinatorial algorithms for unstructured, dynamic and/or adaptive computations.

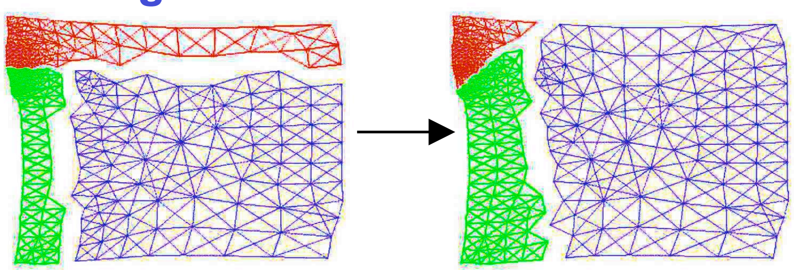
Dynamic Load Balancing



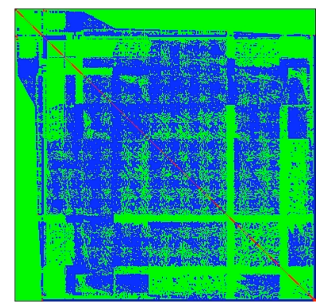
Graph Coloring



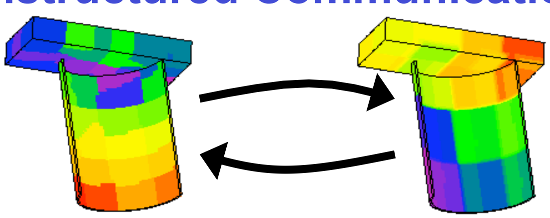
Data Migration



Matrix Ordering



Unstructured Communication

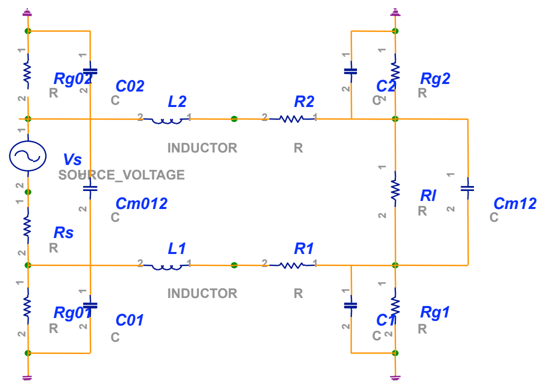


Distributed Data Directories

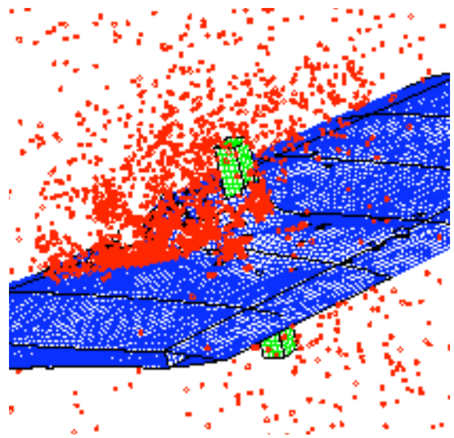
A	B	C	D	E	F	G	H	I
0	1	0	2	1	0	1	2	1

Zoltan's Use in Applications

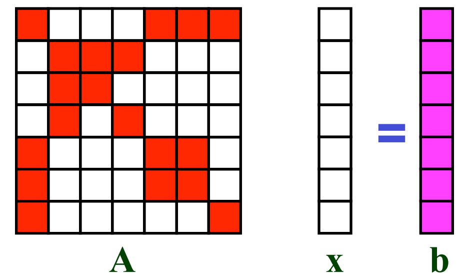
Data-structure neutral design supports many different applications.



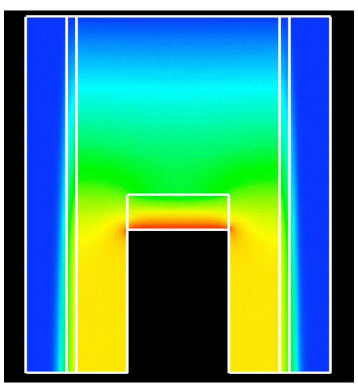
Parallel electronics networks



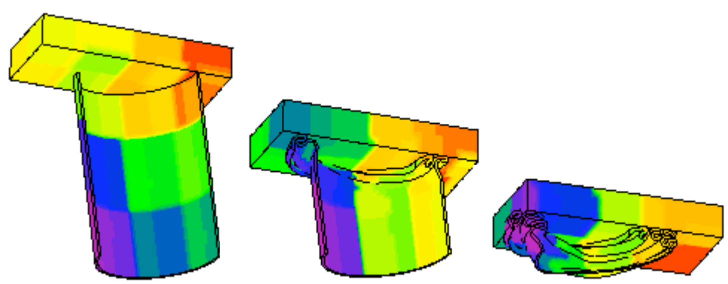
Particle methods



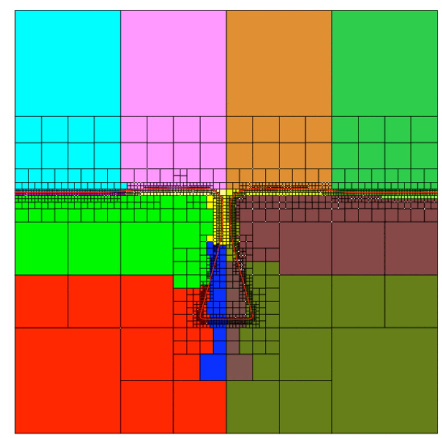
Linear solvers & preconditioners



Multiphysics simulations



Crash simulations



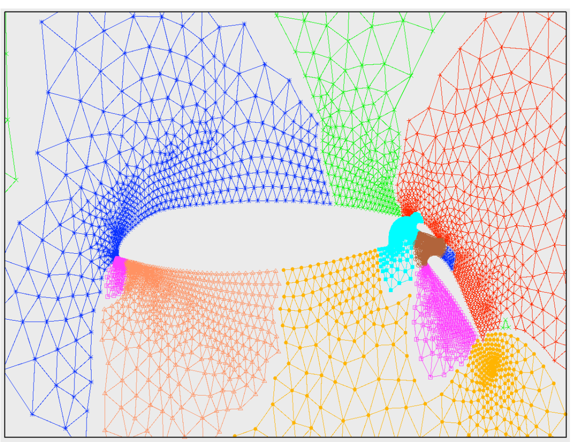
Adaptive mesh refinement

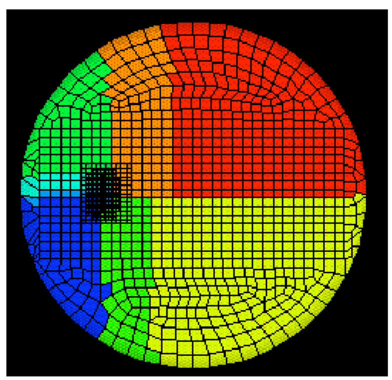
Zoltan's use in large-scale experiments and simulations

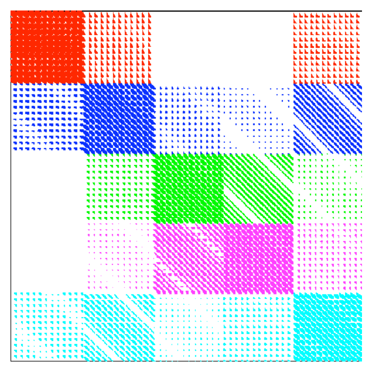
Partitioning Method	Application	Problem Size	Number of Processes	Number of Parts	Architecture	Source
Graph	PHASTA CFD	34M elements	16K	16K	BG/P	Zhou, et al., RPI
Hypergraph	PHASTA CFD	1B elements	4096	280K	Cray XT/5	Zhou, et al., RPI
Hypergraph	Sparta LB algorithms	800M zones	8192	262K	Hera (AMD Quadcore)	Lewis, LLNL
Geometric	Pic3P particle-in-cell	5B particles	24K	24K	Cray XT/4	Candel, et al., SLAC
Geometric	MPSalsa CFD	208M nodes	12K	12K	RedStorm	Lin, SNL
Geometric	Trilinos/ML Multigrid in ALEGRA shock physics	24.6M rows 1.2B non-zeros	24K	24K	RedStorm	Hu, et al., SNL

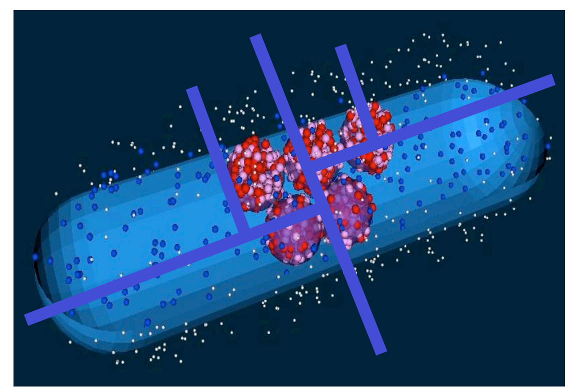
Partitioning and Load Balancing

- Assignment of application data to processors for parallel computation.
- Applied to grid points, elements, matrix rows, particles, ...
- Trade-offs in partitioning and load-balancing algorithms:
  - Quality vs. speed.
  - Geometric locality vs. data dependencies.
  - High-data movement costs vs. tolerance for remapping.



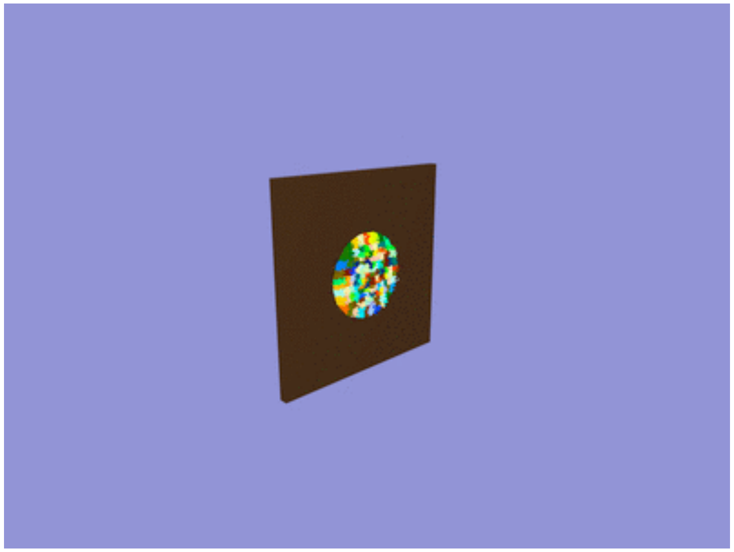






Zoltan's Suite of Partitioning Algorithms

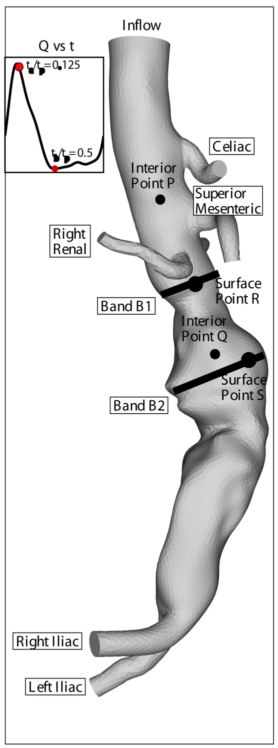
- Geometric methods: Partition based on geometric locality.
  - Parts contain objects that are physically close to each other.
  - Useful for particle methods, adaptive mesh refinement, visualization, contact detection, crash simulations, specialized geometries



***Zoltan's geometric partitioning in SLAC's PIC3P enabled solution of large particle-based problems* (24k CPUs, 750M DOFs, 5B particles). Courtesy of Arno Candel, SLAC, 2009.**

Zoltan's Suite of Partitioning Algorithms

- Topology-based methods: Partition based on connectivity.
  - Parts contain objects that depend on each other.
  - Graph and hypergraph partitioning methods.
  - Useful for matrices, networks, meshes, multiphysics, irregular data.

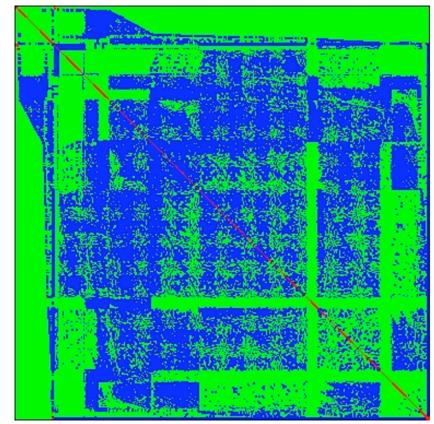


Number of cores	Time (s)	Efficiency
16k	222.03	1
32k	112.43	0.987
64k	57.09	0.972
128k	31.35	0.885

Zoltan's topology-based methods helped achieve strong scalability beyond 128K cores (BG/P) for CFD code PHASTA.
Courtesy of Mark Shephard, RPI.

Zoltan Ordering

- Global ordering produces fill-reducing permutations for sparse matrix factorization.
  - Interfaces to PT-Scotch (Pellegrini, Chevalier; INRIA-LaBRi) and ParMETIS (Karypis et al.; U. Minnesota)

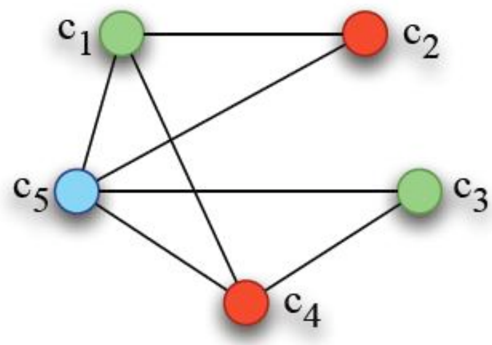


- Local ordering improves cache utilization.
  - Space-filling curve ordering of in-processor data.

Zoltan's local data ordering enabled 13-25% reduction in overall execution time in finite volume climate code FV-MAS. Courtesy of Michael Wolf, SNL.

Grid size	Reorder cells	Reorder cells & edges	Reorder cells, edges & vertices
163842	13.4%	14.8%	15.6%
655362	19.0%	19.4%	17.7%
2621442	23.0%	24.9%	20.6%

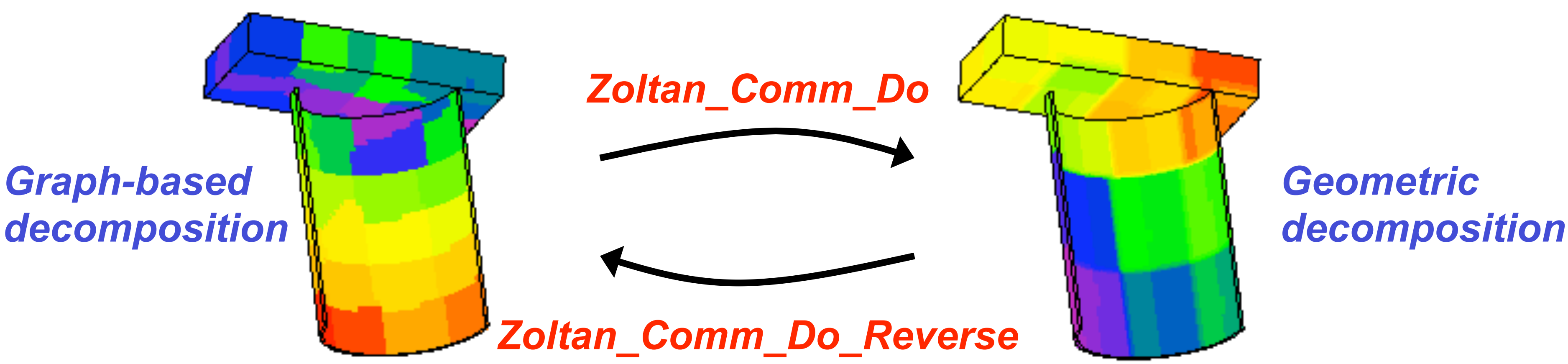
Zoltan Graph Coloring



- Assign colors (labels) to vertices such that neighboring vertices have different colors.
- Parallel distance-1, distance-2 and partial distance-2 graph coloring.
  - Finding independent sets and concurrent computations (e.g., for multithreaded operations)
  - Efficient Jacobian and Hessian calculations (by identifying structurally orthogonal representations of matrices)

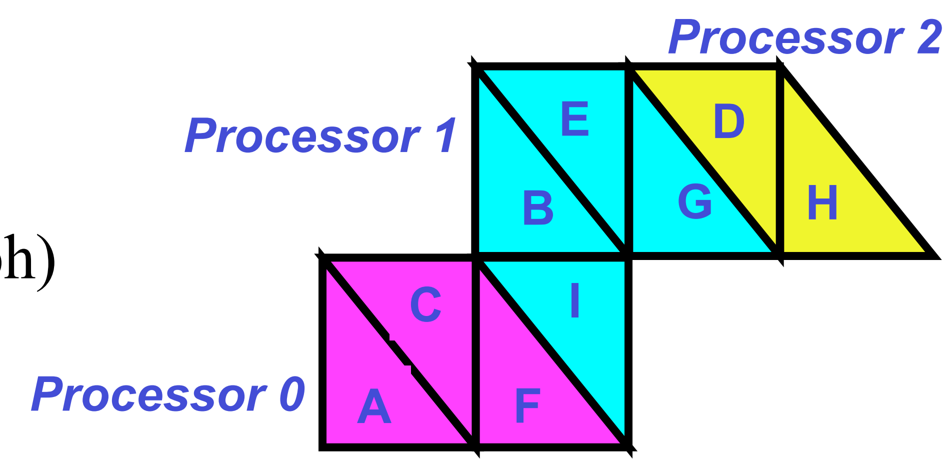
Zoltan Unstructured Communication Package

- Simple primitives for efficient irregular communication.
  - Zoltan_Comm_Create: Generates communication plan.
    - Processors and amount of data to send and receive.
  - Zoltan_Comm_Do: Send data using plan.
    - Can reuse plan. (Same plan, different data.)
  - Zoltan_Comm_Do_Reverse: Inverse communication.
- Used for most communication in Zoltan.
- Exposed through API for application use.



Zoltan Distributed Data Directory

- Allows applications to locate off-processor data.
- Rendezvous algorithm (Pinar, 2001).
  - Directory distributed in known way (hashing) across processors.
  - Requests for object location sent to processor storing the object’s directory entry.
- Used in finite element and particle-in-cell codes (e.g., Aleph) to determine communication patterns.



	Processor 0			Processor 1			Processor 2		
Directory Index →	A	B	C	D	E	F	G	H	I
Location →	0	1	0	2	1	0	1	2	1

Zoltan Software

- Open-source software under LGPL license.
  - http://www.cs.sandia.gov/Zoltan
- Callback-function API:
  - Separates application data from Zoltan data
  - Easy to use for applications; no complicated data structures to build
  - Allows use of Zoltan in wide-range of applications
- Interfaces:
  - C, C++, Fortran90
  - Matrix-based interface through Trilinos
  - Mesh-based interface through ITAPS

Trilinos provides building blocks for application development and research.

physics

L(u)=f
Math. model

$L_h(u_h)=f_h$
Numerical model

$u_h=L_h^{-1} \cdot f_h$
Algorithms

computation

Numerical math
Convert to models that can be solved on digital computers

Algorithms
Find faster and more efficient ways to solve numerical models

Trilinos

discretizations
- Time domain
- Space domain

methods
- Automatic diff.
- Domain dec.
- Mortar methods

solvers
- Linear
- Nonlinear
- Eigenvalues
- Optimization

core
- Matrix/Vector
- Utilities
- Interfaces
- Load Balancing

FASTMath contacts for Trilinos:
Andy Salinger (agsalin@sandia.gov)
Jonathan Hu (jhu@sandia.gov)

Trilinos
Software Infrastructure

- Trilinos Capabilities Areas:
  - Discretizations
  - Linear & Eigen Solvers
  - Nonlinear, Transient & Optimization Solvers
  - Scalable I/O
  - Software Engineering Technologies & Integration
  - Testing, Tools & Interfaces
  - Scalable Linear Algebra
  - Meshes & Load Balancing
- Trilinos is NOT a single monolithic piece of software.
  - Capabilities distributed in individual "packages."
  - Any collection of packages can be combined.
  - Applications don't need all of Trilinos to get things done.

For more information…

- Zoltan website: http://www.cs.sandia.gov/Zoltan
  - Download Zoltan as part of Trilinos or as stand-alone library.
- Trilinos website: http://trilinos.sandia.gov
- Annual Forums:
  - DOE ACTS Tutorial (3rd week in August) at LBL.
  - Annual Trilinos User Group Meeting in November at SNL.

Trilinos Core Packages

Objective	Packages
Parallel/serial Matrix/ Vector classes	**Epetra:** production-ready; C++; double-precision **Tpetra:** next-generation C++; templated scalar & ordinal types **Kokkos:** multicore/GPU node description and operators
Interfaces	**Thyra:** Abstract interfaces to linear algebra **Stratimikos:** Abstract problem description **FEI, Shards:** Finite-element interfaces
Load Balancing, Ordering, Coloring	**Zoltan:** suite of combinatorial algorithms **Isorropia:** Epetra interface to Zoltan
"Skins"	**PyTrilinos:** Python interfaces using SWIG **WebTrilinos**: Web-based interface for testing, experimentation **ForTrilinos:** Fortran interface **Ctrilinos:** C wrappers
C++ utilities	**Teuchos:** Timers, parameter lists, reference-counted pointers; LAPACK/BLAS wrappers **EpetraExt:** transforms; matrix-matrix multiply; transpose **Triutils**: I/O with common matrix formats

Trilinos Discretizations and Methods

Objective	Package(s)
Mesh management	**STKMesh:** Flexible mesh database **Pamgen**: In-line mesh generation **Mesquite:** Mesh-quality improvement; r-refinement
Discretization	**Intrepid:** discretization for general FEM, FV, & FD cell types **Sundance:** Finite element method; declarative programming **Phalanx:** Field-evaluation kernel
Time Integration	**Rythmos:** backward/forward Euler, Runge-Kutta, BFD
Automatic differentiation	**Sacado:** AD at element level via templating; forward/reverse/Taylor-polynomial modes
Mortar methods	**Moertel:** nonconforming mesh tying and contact formulations

Trilinos Solvers

Objective	Package(s)
Iterative linear solvers	**AztecOO:** Krylov subspace solvers: CG, GMRES, Bi-CGSTAB; Incomplete factorization preconditioners **Belos:** Block-based solvers; recycling solvers; templated C++ **Komplex:** solves complex-valued linear systems via equivalent real-valued systems
Direct sparse linear solvers	**Amesos/Amesos2**: Interface to direct solvers KLU, UMFPACK, SuperLU, MUMPS, ScaLAPACK
Direct dense linear solvers	**Epetra, Teuchos, Pliris**
Iterative eigenvalue solvers	**Anasazi:** Block-based Krylov-Schur, Davidson, LOBPCG
ILU-type preconditioners	**AztecOO** **IFPACK/Ifpack2:** Overlapping Schwarz
Multilevel preconditioners	**ML:** Smoothed aggregation; multigrid; domain decomposition
Block preconditioners	**Meros, Teko**: for coupled simultaneous solution variables
Nonlinear system solvers	**NOX:** Broyden, Newton, Tensor methods **LOCA:** Continuation algorithms
Optimization (SAND)	**MOOCHO, Aristos:** Reduced and full-space SQP **TriKota:** Interface to DAKOTA toolkit
Stochastic PDEs	**Stokhos:** intrusive stochastic Galerkin uncertainty quantification

Full Vertical Solver Coverage

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 $u \in U \ni \frac{\partial F}{\partial x}$ s	**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 $F(x) \in \Re^m \to \Re$ Solve $F(x) = 0 \quad x \in \Re^n$	**NOX**
Linear Problems **Linear Equations:** **Eigen Problems:**	Given Linear Ops (Matrices) $A, B \in \Re^{m \times n}$ Solve $Ax = b$ for $x \in \Re^n$ Solve $A\nu = \lambda B\nu$ for (all) $\nu \in \Re^n, \quad \lambda \in$	**AztecOO** **Belos** **Ifpack, ML, etc...** **Anasazi**
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**

Sensitivities
(Automatic Differentiation: Sacado)