SAND2011-8187 C
SAND2011-8187C

Next-generation iterative solvers for next-generation computing: Anasazi and Belos

Mark Hoemmen `mhoemme@sandia.gov`

Sandia National Laboratories

02 Nov 2011

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.

Who am I?

- Postdoc at Sandia National Laboratories
  - Graduated UC Berkeley spring 2010
- Research: "Scalable algorithms"
  - Interactions between algorithms and computer architectures
- Trilinos developer since Spring 2010
  - New, fast, accurate block orthogonalization (TSQR)
  - New iterative linear solvers in progress
  - Sparse matrix I/O, utilities, bug fixes, and consulting
- Trilinos packages I've worked on:
  - Anasazi, Belos, Kokkos, Teuchos, Tpetra

List of contributors

- Anasazi and Belos share many contributors
  - Common initial design
  - Anasazi motivated Belos in part
- Common lead:
  - Heidi Thornquist
- Contributors:
  - Chris Baker, David Day, Mike Heroux, Ulrich Hetmaniuk, Sarah Knepper, Rich Lehoucq, Mark Hoemmen, Vicki Howle, Mike Parks, Kirk Soodhalter, …

“State of the union”: Outline

- Motivations for the two packages
  - Application-aware, architecture-aware algorithms
  - Adapt quickly to rapidly evolving computer architectures
- New features (since last TUG)
  - Including new solvers!
- Design evolution discussion: Help Anasazi & Belos…
  - Track architecture evolution
  - Support new solver algorithms

Support algorithms that are…

- Architecture-aware
  - "Flops are cheap, bandwidth is money, latency is expensive"
    – Kathy Yelick
  - Favor "block" kernels that amortize data movement cost over several vectors
    - Sparse matrix times multiple vectors
    - Block vector operations
- Application-driven
  - Many apps don't just solve one linear system
  - Apps really solve "block" problems…
    - Eigenvalue clusters
    - $AX = B$
    - $(A + \Delta A_j)\, X_j = B + \Delta B_j$
  - Use cases:
    - Nonlinear solvers
    - Time evolution
    - Parameter studies
- *Convergence* of computational kernels and algorithms

Sandia National Laboratories

Abstract interface lets solvers track architecture evolution

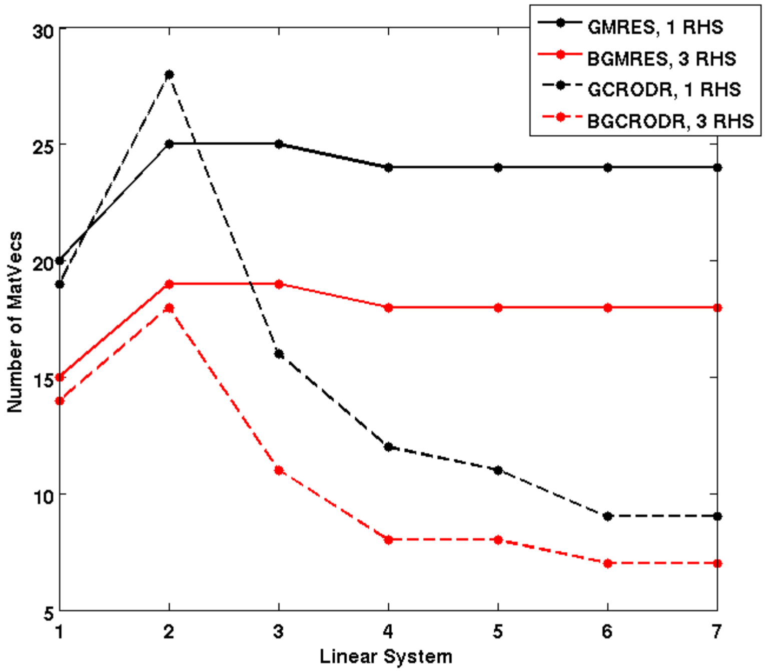
- Older packages (Aztec(OO), ARPACK)
  - “Reverse communication” interface
  - Constrains vector (& matrix) representation
- Anasazi and Belos
  - Only constrains *algebra* (interface) of operators and vectors
  - Does *not* constrain data representation
- Decoupling from data representation
  - Solvers work with your favorite linear algebra library
    - Epetra, Tpetra, Thyra, yours (if you wrap it)
  - Enables evolution to different node architectures and programming models
    - Optimal data placement critical for performance
    - Best placement depends on the hardware

Trilinos

New solvers and features

Block Recycling GMRES (Block GCRO-DR)

- Algorithm: Kirk Soodhalter (Temple U, Daniel Szyld)
- Belos implementation: Kirk S. and Mike Parks
- Reuse basis from previous solves to accelerate sequences of solves
- Example: Tramonto
  - Fluid density functional theory
  - Hard spheres w/ electrostatics and attractions
  - Newton iteration: 7 solves
  - Savings:
    - 1 RHS: 60 matvecs (36%)
    - 3 RHS: 50 matvecs (40%)



LSQR: Least-squares solver (1 of 2)

- Algorithm: Michael Saunders (Stanford)
- Belos implementation:
  - Sarah Knepper (Emory, now Intel) and David Day
- LSQR solves
  - Nonsymmetric linear systems
  - Linear and damped least squares
- Algorithmic features
  - Detects incompatible Ax=b; returns least-squares solution
  - Tolerates singular matrix A; works with nonsquare A
  - Computes sparse SVD: sharp condition number bounds
  - Fixed memory footprint (but more matvecs than GMRES)

Sandia National Laboratories

LSQR: Least-squares solver (2 of 2)

- Use case: Adaptive-precision solver
  - Mixed & arbitrary precision an important Belos motivation
  - Prefer single to double precision
    - Memory bandwidth and memory per node constrained on modern computers
  - But A may be singular in single, not in double
  - while(cond(A) > 1 / eps(prec)) { increase prec, solve again }
- Other applications
  - Nonlinear least squares (trust region search)
  - Certain inverse problems: $\min || b – Ax ||^2 + \mu ||Lx||^2$
- Software notes
  - Requires transpose: first Belos solver that does!
  - This helped us discover and fix Belos' Epetra wrappers

Sandia National Laboratories

MINRES: Linear solver

- Algorithm: Paige and Saunders
- Belos implementation: Nico Schlömer
  - With help from Heidi Thornquist and Mark Hoemmen
- Solves symmetric indefinite linear systems
  - Fixed memory footprint
- Result of Nico's TUG 2010 presentation!
  - Nico: "You can see CG deflating the negative eigenvalues…"
  - me: [cringes visibly]
  - Inspired Nico to contribute MINRES implementation

Sandia National Laboratories

Faster orthogonalizations, more easily available

- Tall Skinny QR (TSQR) orthogonalization method
  - 2008 UC Berkeley tech report, SC09, IPDPS 2011, …
  - O(1) reductions, independent of number of vectors
- Now works with any Tpetra type on CPU node
  - Kokkos Node = TPINode, TBBNode, SerialNode
  - Algorithm specialized for Kokkos node type
- Also works with Epetra, if Trilinos built with Tpetra
- In Belos: Available via OrthoManagerFactory
  - Solvers no longer have to construct OrthoManager
  - Factory handles interpreting parameters
    - Sublist “Orthogonalization Parameters”
  - Available in GCRODR, soon in other GMRES variants

Sandia National Laboratories

Design evolution discussion

Design evolution

- Refactor solvers' interface to linear algebra?
  - Do Anasazi and Belos need fused computational kernels?
- Improve support for inner-outer iterations?
- Improve robustness to effects of hybrid parallelism?

Fuse computational kernels?

- Anasazi & Belos currently assume separate kernels
  - One kernel = one linear algebra library routine call
  - Vector ops and matrix-vector ops are separate
- Examples of fused kernels:
  - w = A*x, alpha = dot(w,x)
  - w = A*x, z = A^T * y
- Almost always good or harmless for performance
  - Avoid overhead of starting & stopping tasks
  - Increase task duration ➔ maximize data locality
- How would this change solvers?
  - Solver code changes, but algorithms don't (much)
  - Low-risk evaluation using Chris Baker's Tpetra::RTI CG
  - No change to user interface, only to linear algebra interface

Sandia National Laboratories

Improve support for inner-outer iterations?

- Currently: Outer solver treats inner as black box
- Some algorithms want communication between inner and outer solves
  - Example: inexact Krylov (Szyld et al.)
    - Outer solver adjusts inner tolerance based on outer $||r_k||$
  - Example: Fault-Tolerant GMRES (Heroux, Hoemmen et al.)
    - Inner solve events may affect outer solve behavior
- Can we support this without rewriting solvers (much)?

Improve robustness to effects of hybrid parallelism?

- Thread parallelism may not be deterministic
- Parallel BLAS & LAPACK may give different results on different MPI processes
- Anasazi & Belos expect same evaluation of projected (small dense) problem on different processes
- “Continuous” perturbation affects discrete decisions
  - Count of eigenvalues in a cluster
  - Convergence criteria for linear solves
- If some processes go on and others stop:
  - Crash or deadlock
- To fix: No hard math, but redesign of all “parallel decisions” and continuous → discrete transitions

Trilinos

Any questions?

Extra Slides

Design evolution (extra)

- Leave reduction results on the compute device?
  - Current interface returns scalar results from GPU to CPU
  - Instead, could leave results on GPU, fire kernels asynch.
  - Carter Edwards' Gram-Schmidt prototype (ValueView)
  - Solver code changes a LOT; algorithms may too
    - Can't evaluate convergence tests on the GPU
    - Batch up several iterations
  - Not so effective with MPI and multiple GPUs
    - Must communicate the reduction results anyway
    - Can they go straight from the GPU to the network interface

Full Vertical Solver Coverage

Problem	Formulation		Package
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 $u \in U \ni \frac{\partial F}{\partial x}$ 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 $F(x) \in \Re^m \rightarrow \Re$ 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**

Belos

- Next-generation linear iterative solvers

- Decouples algorithms from linear algebra objects
  - Better than “reverse communication” interface of Aztec
  - Linear algebra library controls storage and kernels
  - Essential for multicore CPU / GPU nodes
- 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 methods: Block GMRES and Block CG
  - Recycling solvers: GCRODR (GMRES) and CG
  - “Seed” solvers (hybrid GMRES)
  - Block orthogonalizations (TSQR)
- Supports arbitrary and mixed precision, and complex

Developers: Heidi Thornquist, Mike Heroux, Mark Hoemmen, Mike Parks, Rich Lehoucq

Sandia National Laboratories

Anasazi

- Next-generation iterative eigensolvers

- Decouples algorithms from linear algebra objects
  - Better than “reverse communication” interface of ARPACK
  - Linear algebra library controls storage and kernels
  - Essential for multicore CPU / GPU nodes
- Block eigensolvers for accurate cluster resolution
- Can solve
  - Standard ($AX = \Lambda X$) or generalized ($AX = BX\Lambda$)
  - Hermitian or not, real or complex
- Algorithms available
  - Block Krylov-Schur (most like ARPACK’s IR Arnoldi)
  - Block Davidson
  - Locally Optimal Block-Preconditioned CG (LOBPCG)
  - Implicit Riemannian Trust Region solvers
  - Advanced (faster & more accurate) orthogonalizations

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