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

Mark Hoemmen mhoemme@sandia.gov

Sandia National Laboratories

02 Nov 2011

Unclassified, unlimited release.
SAND # 2011-8187 C

Sandia is a multiprogram 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

Who am I?

- Postdoc at Sandia National Laboratories
  - Graduated UC Berkeley March 2010
- Research: “Scalable algorithms”
  - Interactions between algorithms and computer architectures
- Trilinos developer since Spring 2010
  - New, fast, accurate block orthogonalization (TSQR)
  - New communication-avoiding & fault-tolerant solvers
    - Prototyped & running, not in Belos yet
  - Sparse matrix I/O, utilities, bug fixes, and consulting
- Trilinos packages I’ve worked on:
  - Anasazi, Belos, Kokkos, Teuchos, Tpetra

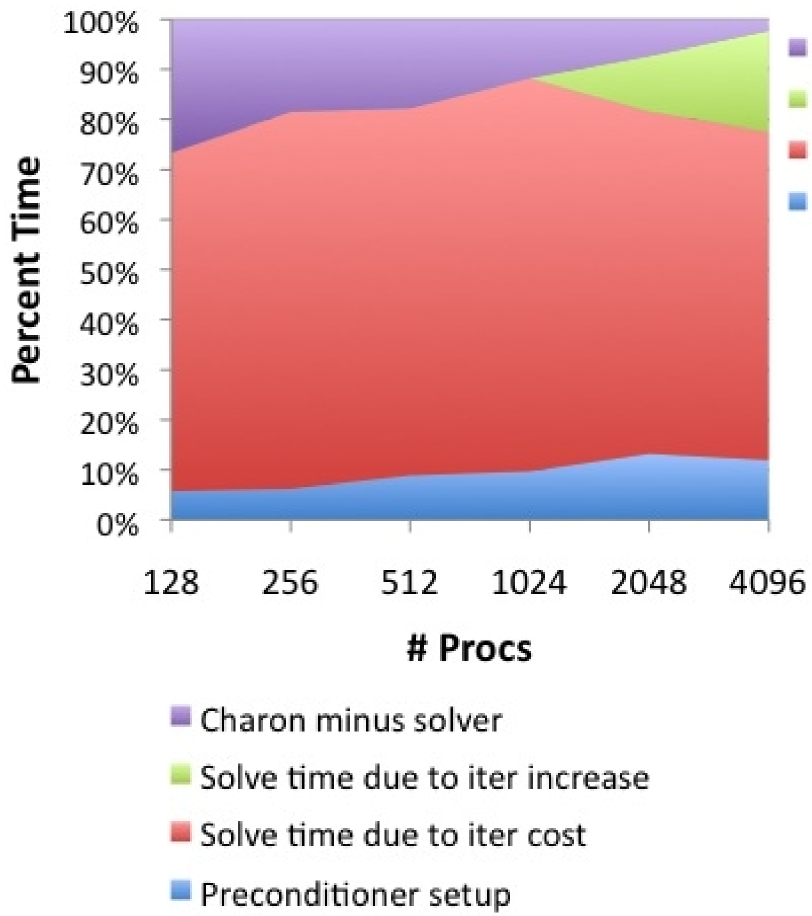
List of contributors

- Anasazi and Belos share many contributors
  - Anasazi came first
  - Belos shared design features & motivations
- 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, …

Outline

- Motivations for Anasazi & Belos
  - Why are iterative solvers still hard?
  - Why block solvers?
  - Why decouple solvers from the linear algebra library?
  - Why are they templated on the Scalar type?
- Application highlights
- New solvers & features (since last TUG)
- Research & development efforts
- Future (ongoing) work

Why are Ax=b & Ax=Λx still hard?



- Often ≥ 80% runtime
- Dominate runtime of higher-level algorithms
  - Nonlinear solves, optimization, statistics, …
- Harder to optimize
  - Bandwidth-bound
  - More communication
- Why we care about…
  - New algorithms & kernels
  - Software flexibility to develop & deploy them

Sandia National Laboratories

Why block solvers?

- "Block" solvers:
  - Resolve clusters of eigenvalues
  - Solve several right-hand sides at once
- Architecture-aware (= "avoid data movement")
  - "Flops are cheap, bandwidth is money, latency is expensive"
    - – Kathy Yelick (LBNL & UC Berkeley)
  - Standard Krylov kernels dominated by data movement costs
  - Favor "block" kernels that amortize costs over many vectors
- Application-driven (= "rarely just one linear system")
  - Needed to resolve tightly clustered eigenvalues
  - Block eigensolver → block linear solver (shift & invert)
  - Parameter studies, robustness, uncertainty, …
- Lucky convergence of architecture and application!

Sandia National Laboratories

Why decouple solvers from the linear algebra library (LAL)?

- "Any problem in computer science can be solved with another level of indirection."
  - Butler Lampson, 1993 Turing Award lecture
- Rapid evolution of computer architectures
  - LAL architects & performance tuners must track them
  - Numerical algorithm developers != performance tuners
  - Free the former to focus on algorithmic evolution
- Data placement crucial for performance
  - LAL must be free to store data how it likes
  - Solvers only interact with data through a few kernels

Anasazi & Belos decoupled from linear algebra library

- Previous packages (AztecOO, ARPACK) were not
  - “Reverse communication” interface, which means here:
    - Decoupled from operator representation
    - Still constrains vector representation
- Anasazi and Belos only constrain interface
  - Compile-time polymorphic “traits” interface
  - Interface cost is at most one function call
  - Solvers work with any linear algebra library
    - Epetra, Tpetra, Thyra, …, yours (wrapped)

Why are Anasazi & Belos templated on the Scalar type?

- Arbitrary-precision algorithms
  - Some problems need extra precision
  - We can do CG & GMRES with
    - double-double (128 bits), quad-double (256 bits), …
  - Wish list: fully templated LAPACK
- Mixed-precision algorithms
  - Use the least precision necessary (e.g., float vs. double)
  - Enable new algorithms that
    - Use lower precision most of the time
    - Use higher precision selectively
  - Save bandwidth & memory
- Avoid code duplication

Sandia National Laboratories

Application highlights

Anasazi application highlights

- Themis: Large data set analysis tool
  - Canonical Correlation Analysis
  - Computed by eigen{value + vector} solve(s)
  - Anasazi provides efficient parallel implementation
- Schrödinger's equation solver
  - Part of QCAD (Quantum CAD) LDRD
  - Equations set up in Albany
  - Anasazi accessed through LOCA
- Block Krylov-Schur with > 2^40 unknowns
  - 1,728,684,249,600 (> 1 trillion!) unknowns
  - k-eigenvalue problem in Denovo (reactor design)
  - 200K cores of Jaguar

Sandia National Laboratories

Belos application highlights

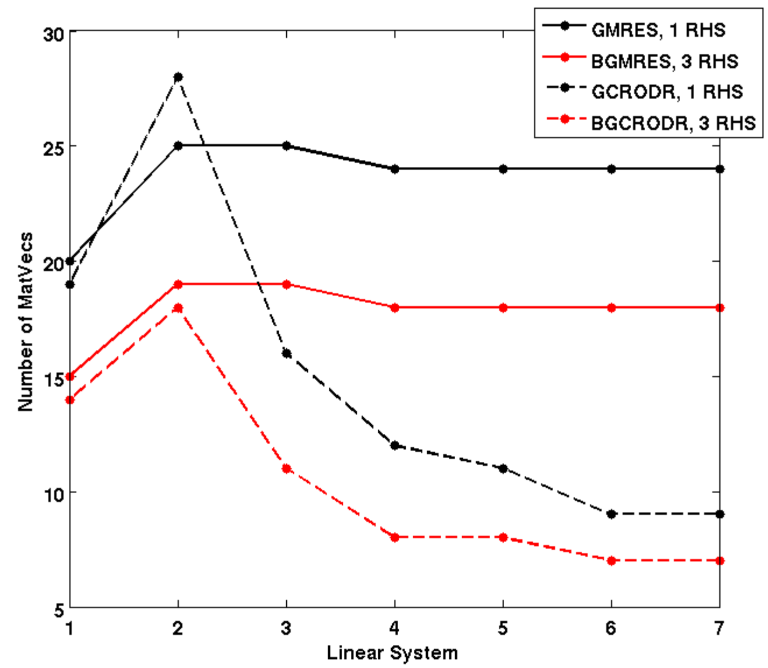
- GLIMMER Community Ice Sheet Model
  - Flexible GMRES inner-outer iteration, driven by NOX
  - Trilinos driven by Andy Salinger's Piro package
  - Fortran 95 (they have custom Trilinos wrappers)
  - Provided feedback that helped us fix a performance bug
    - Teuchos::TimeMonitor::summarize()
- LifeV finite-element library
  - Fruitful collaboration with EPFL visitors
- Belos already being integrated into more codes
  - Epetra → Tpetra requires AztecOO → Belos
  - Expect heavier Belos use as Tpetra-based stack matures

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%)



Sandia National Laboratories

LSQR: Least-squares solver (1 of 2)

- Algorithm:
  - C. C. Paige & M. A. 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(precision)):
    - Increase precision
    - Solve again
- 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

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 Tpetra on any 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
  - Decouples solvers from orthogonalization setup
  - Factory handles interpreting parameters
    - Sublist “Orthogonalization Parameters”
  - Available in GCRODR, soon in other GMRES variants

Research & development efforts

Communication-avoiding solvers

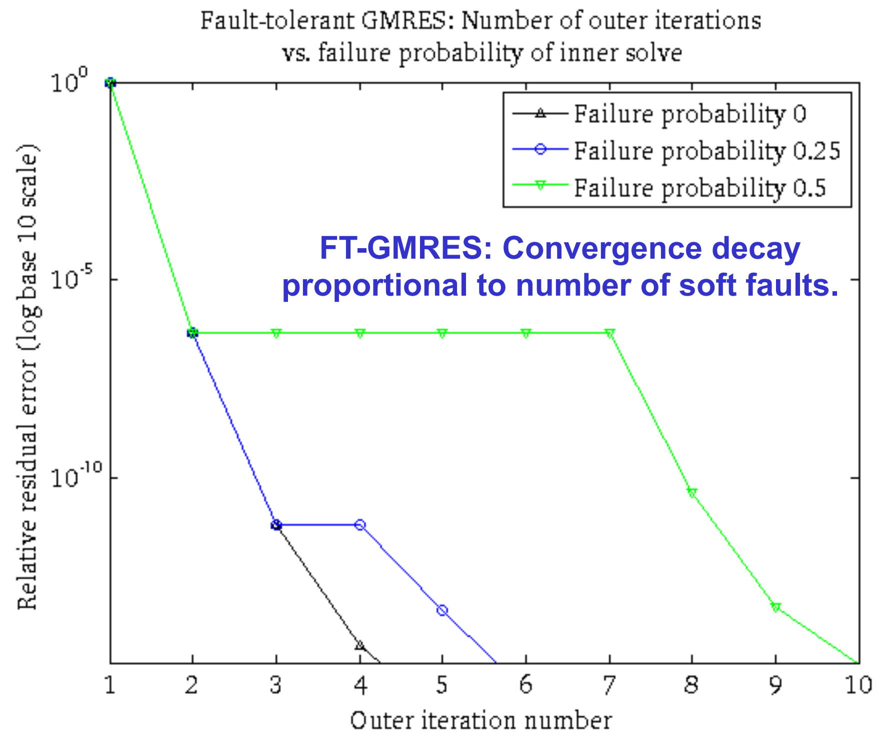
- "Communication" = data movement
  - Between levels of memory hierarchy (bandwidth)
  - Between parallel processors (latency)
  - Slow & getting slower exponentially relative to flops
- Standard Krylov methods are communication-bound
- "Communication-avoiding" (CA) solvers:
  - Use different kernels that communicate less
  - Details: Hoemmen 2010 (PhD thesis), …
- Trilinos prototype of CA-GMRES
  - Built on Tpetra and Belos; already getting speedups
  - ~ 3 weeks of work to deploy in Belos
- Long-term collaboration with UC Berkeley and others
  - Kernels and kernel optimizations
  - New CA algorithms & lower bounds theory

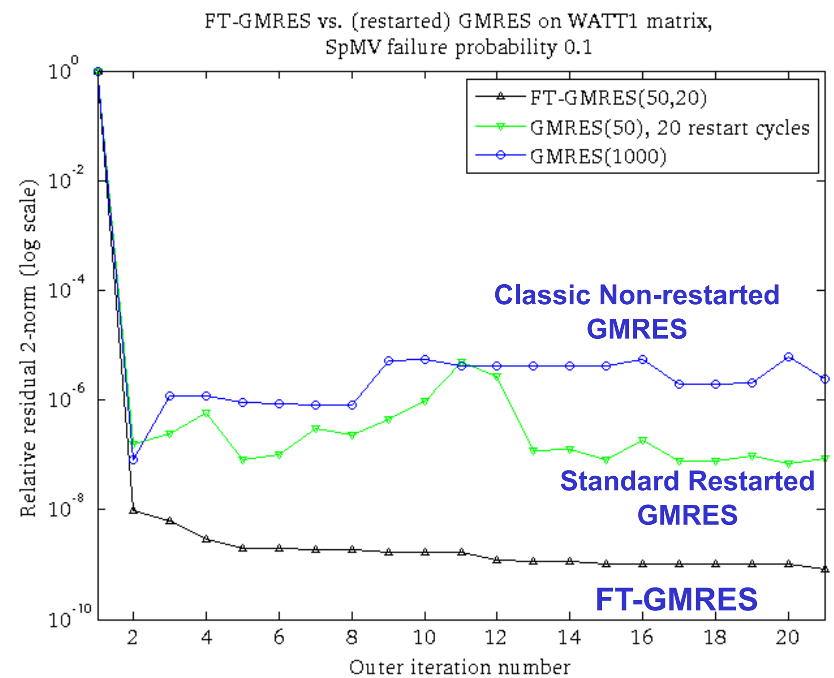
Fault-tolerant solvers

- Exascale systems will be less reliable
  - Including incorrect data and computations
  - Reliability has energy and performance cost
- Iterative solvers are…
  - Sensitive to unreliable data and computations
  - Faults may cause incorrect results *undetectably*
- “Selective reliability” enables new solvers
  - System exposes reliability tradeoffs
  - Algorithm identifies what *must* be reliable
  - This requires new iterative solver algorithms!
- Fruitful collaboration with systems researchers
  - Sandia’s “9 Lives” Group (Patrick Bridges, Kurt Ferreira)

Fault-Tolerant GMRES

- An inner-outer iteration
  - Based on Flexible GMRES
  - Inner solver “preconditions” outer solver
  - Inner solver runs unreliably
  - Outer solver runs reliably
- Advantages
  - Reuse any existing solver stack as “inner solver”
  - Most time spent in cheap unreliable mode
  - Faults only delay, don’t prevent convergence
  - Can exploit fault detection if available, but not necessary





Future (ongoing) work

Future (ongoing) work

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

Fuse computational kernels?

- Anasazi & Belos currently assume separate kernels
  - One kernel = one linear algebra library routine call
- Examples of fused kernels:
  - w = A*x, alpha = dot(w,x)
  - w = A*x, z = A^T * y
- Good or harmless for performance
  - Avoid overhead of starting & stopping tasks
  - Increase task duration ➔ maximize data locality
  - May allow launching kernel(s) asynchronously
- How would this change solvers?
  - Solver code changes, but algorithms don't (much)
  - Low-risk evaluation using Chris Baker's Tpetra::RTI CG

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)?

Sandia National Laboratories

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

Sandia National Laboratories

Summary

- Linear algebra is still hard
- Advantages of Anasazi & Belos
  - Block algorithms desired by applications & perform well
  - Solvers decoupled from matrix & vector storage layout
  - Mixed- & arbitrary-precision algorithms through templating
  - Can solve problems with > 2 billion unknowns
- Critical for manycore performance
  - Fully compatible with Tpetra & Epetra stacks
  - Simplifies Epetra → Tpetra transition
- Advanced new algorithms

Sandia National Laboratories

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

Abstraction lets solvers track architecture evolution

- LAL (*not* solvers) carries evolution burden
  - Solver developers often not performance tuners
  - They can focus on algorithmic evolution
- LAL (*not* solvers) controls all…
  - Data placement
    - Needed for accelerator architectures (e.g., GPUs)
    - Performance critical on multicore CPUs
  - Intranode (thread) & internode (MPI) parallelization
    - Solver developers don't need to write OpenMP, CUDA, …
- Disadvantages
  - LAL interface constrains cross-kernel optimizations

Sandia National Laboratories