Exceptional service in the national interest

Sandia National Laboratories

Production Implementations of Pipelined & Communication-Avoiding Iterative Linear Solvers

Mark Hoemmen (SNL, UUR: SAND2018-2310 C)
& Ichitaro Yamazaki (UTK)

SIAM Parallel Processing, March 09, 2018

U.S. DEPARTMENT OF ENERGY
NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

DOE's ECP PEEKS

E. Boman & M. Hoemmen (SNL)
J. Dongarra, H. Anzt, S. Tomov, & I. Yamazaki (UTK)

Sandia National Laboratories

- Pipelined & communication-avoiding Krylov solvers
  - Hide (overlap) or avoid (do less) communication (e.g., all-reduce)
  - May perform better, especially at large scales
- We want to make these solvers available in Trilinos
  - Goal: perform "well" for ECP applications on exascale computers
  - Also should be useful for non-exascale Trilinos users
- This talk
  - Is about resulting Trilinos software development challenges
  - Is NOT about algorithms or their performance

What is Trilinos?

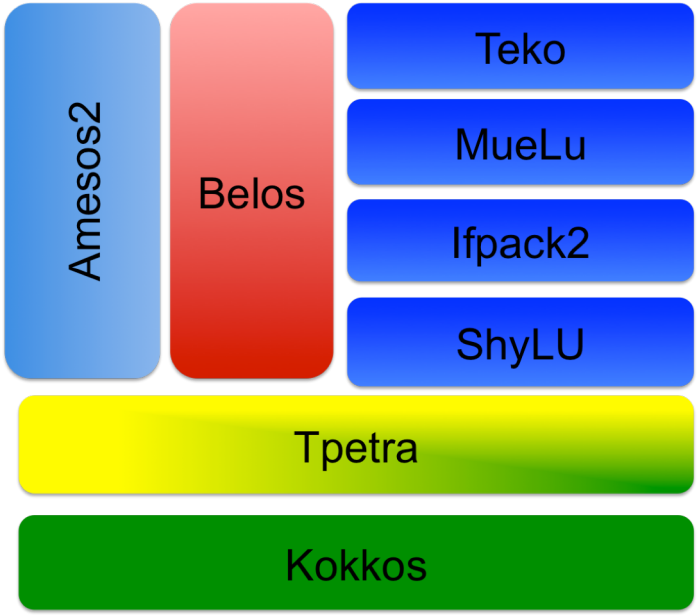
Sandia National Laboratories

- Parallel math libraries for science & engineering applications
  - Parallel programming models
  - Sparse linear algebra
  - Linear & nonlinear solvers
  - Optimization algorithms
  - Space & time discretizations
- Mostly C++ with some C and Fortran
- Many users inside & outside Sandia
- Must work on many different platforms
  - CPUs: x86, KNL, POWER, ARM, …
  - GPUs: NVIDIA, later AMD
  - Future architectures (e.g., exascale)
- github.com/trilinos/Trilinos

Trilinos' linear solvers

Sandia National Laboratories

- Iterative (Krylov) solvers (Belos)
  - CG, GMRES, BiCGStab, TFQMR, recycling methods
- Linear algebra operations
  - Tpetra: Sparse graphs/matrices, dense vectors, parallel solve kernels, communication & redistribution
  - Teuchos: BLAS/LAPACK
- Sparse direct solvers (Amesos2)
- Direct+iterative solvers (ShyLU)
- Algebraic iterative solvers (Ifpack2)
  - Jacobi, SOR, polynomial, incomplete factorizations, additive Schwarz
- Algebraic multigrid (MueLu)
- Segregated block solvers (Teko)



Goal: “Productionize” solvers

Sandia National Laboratories

- Make pipelined & communication-avoiding iterative linear solvers available in Belos for Trilinos users
  - Must build & pass tests on all supported platforms
- Available to users via run-time choice (input deck)
  - Users don’t need to change their code
  - Plugging solvers through a custom solver “factory”
  - Not just dumping a class or function into the code repository

Software challenges

Sandia National Laboratories

1. Trilinos' iterative linear solvers package makes it hard to add new linear algebra operations

2. Trilinos must support many different build configurations
   - Older MPI versions may lack features needed for pipelined Krylov solvers
   - Default MPI implementation settings may not make progress on nonblocking collectives, thus taking away benefit of pipelined Krylov methods

Trilinos' Belos package

Sandia National Laboratories

- Trilinos' iterative linear solvers live in the Belos package
- Belos was written in the mid-2000's, to support Anasazi (iterative eigensolvers package)

- Belos works for any linear algebra (LA) implementation
  - Belos defines a fixed set of ops on Vectors & Linear Operators (matrices & preconditioners): e.g., dot, norm, axpy, apply(X,Y)
  - Fixed set of LA ops defined via (C++) traits classes

  - Belos' solvers are templated on Vector & Linear Operator & invoke LA ops by using traits classes
  - Belos provides specializations of traits for Trilinos' native LA types (e.g., Tpetra)

Belos solver interface

```
// Create a Belos::SolverManager for CG
// rcp / RCP are Trilinos' std::shared_ptr (Belos only needs C++98)
auto belosSolver = rcp (new Belos::PseudoBlockCGSolMgr<Scalar, Vec, LinOp>);

// Create and Set problem
typedef Belos::LinearProblem<Scalar, Vec, LinOp> linear_problem_type;
RCP<linear_problem_type > lp (new linear_problem_type (A, X, B));

belosSolver->setProblem (lp);
belosSolver->setParameters (params); // e.g., iteration count, convergence tolerance

// Solve the system
Belos::ReturnType belosResult = belosSolver->solve ();
```

- Same solver manager may be reused for multiple solves;
  e.g., for different b or A.
- A, b, x may be templated, e.g., with scalar, global/local ordinal, and node type

Belos solver implementation

```
typedef MultiVecTraits<Scalar, Vec> MVT;

RCP<Vec> P_;
RCP<Vec> AP_;

// Compute AP = A*P
lp_->applyOp( *P_, *AP_ );

// Compute P^T * A P
MVT::MvTransMv( one, *P_, *R_, alpha );
```

- If users want Belos to work for their own LA types, they must write their own specializations of traits classes

Challenge 1: New LA operation breaks build

Sandia National Laboratories

- What if we need new LA ops?
  - Pipelined Krylov: Nonblocking dot product
  - Communication-avoiding Krylov: Matrix powers kernel, TSQR
- Can't add new LA ops to Belos without breaking build!
  - OK for Trilinos' native LA
    - Belos just changes its traits class specializations
  - NOT OK for users' own LA
    - Users have their own Belos traits specializations
    - They would need to change their code to add new LA ops

Run-time vs. compile-time

Sandia National Laboratories

- **Mark**: “Not a C++ problem, but a design problem”
- Belos could have used run-time polymorphism (inheritance)
  - Adding new LA ops through “mix-in” classes without breaking backwards compatibility for libraries that lack them
- Belos chose compile-time polymorphism for historical reasons
  - Belos is in 1st generation of Trilinos packages using templates
  - Early C++ adopters for math codes worried about run-time overhead
  - C++ templates promised zero overhead
  - Trilinos developers have more experience w/ templates now
    - Tiny virtual method dispatch overhead vs. MPI communication
- Goal: no code changes for Belos users who want to use our new solvers

Traits classes too rigid here

Sandia National Laboratories

- One solver implementation for all LA ops, and one traits class contains all LA ops; but
  - Some solvers need specialized ops
  - Users or third-party libraries may have optimized entire solvers for specific LA; Belos users want to access them
- Solution: extend Belos to support LA-specific solvers
  - Belos::SolverFactory already takes solver name at run time & returns instance of the desired solver
  - NEW interface to inject a "**custom** solver factory" at run time
    - SolverFactory class templated on Vector & LinearOperator
      ➔ custom factory is specific to those types
    - custom solver needs to work for one LA
      ➔ they can code directly to that LA & use whatever ops they want
  - Solves a more general problem than pipelined & CA Krylov

Belos factory interface

```
// Create a Solver Manager
    Belos::SolverFactory<Scalar, Vect, LinOp> factory;
    RCP<solver_manager_type> solverManager = factory.create (args.solverName, params);

// Solve the system
    Belos::ReturnType belosResult = solverManager->solve ();
```

Behind the scenes:

```
// Create a Custom Solver Factory
    RCP<custom_factory_type> customFactory;
    customFactory = rcp (static_cast<custom_factory_type*> (new PeeksKrylovFactory<> ()));

// Add the Custom Factory
    factory.addFactory (customFactory);
```

Belos factory interface

```
// Create a Solver Manager
    Belos::SolverFactory<Scalar, Vect, LinOp> factory;
    RCP<solver_manager_type> solverManager = factory.create (args.solverName, params);

// Solve the system
    Belos::ReturnType belosResult = solverManager->solve ();
```

Behind the scenes:

```
void createSolver (const std::string& solverName)  {
   if (solverName == "PeeksCgPipeline") {
     this->solver_ = Teuchos::rcp( new CgPipeline<Scalar, LocalOrd, GlobalOrd, Node> () );
   } else if ( …
}
```

New Op for nonblocking all-reduce

Sandia National Laboratories

- Tpetra's interface to this new specialized op:
  - auto request = idot(&result, x, y); // ← MUST NOT BLOCK
  - /* … do other stuff ... Then */
  - request->wait();
- MPI-3 (2012) added support for nonblocking collectives
  - MPI_Iallreduce: nonblocking version of MPI_Allreduce
- What if Trilinos was built with MPI < 3?
  - Capture (&result, x, y) in a closure (C++11 lambda) that does blocking dot product; don't invoke closure yet
  - request->wait() just invokes the closure as std::function
- We write the solver once; & it works for all MPI versions

Challenge 2: nonblocking progress

Sandia National Laboratories

- "Nonblocking" ➔ return immediately after being called
- MPI could just defer all communication until MPI_Wait
  - MPI may only send/receive inside MPI functions
  - For asynchronous progress, must enable MPI_THREAD_MULTIPLE support & possibly also "progress thread" options at MPI build time
- Problems:
  - Users / sysadmins, not Trilinos, pick build MPI options
  - MPI_THREAD_MULTIPLE & progress thread incur overhead
  - Would we need to poll manually?
    - Paul Eller (UIUC): PETSc's implementation of pipelined Krylov needed manual MPI polling embedded inside the sparse matrix-vector multiply kernel in order for MPI_Iallreduce to be effective (!)

MPI progress: “work in progress”

Sandia National Laboratories

- Not sure what to do about this, yet
  - Manual polling? Impossible on GPUs, invasive in code
  - Programming model mismatch
    - Pipelined Krylov methods really want a dataflow model
    - MPI historically resisted “active messages” (that run a function asynchronously when I receive data from another process)
- MPI_THREAD_MULTIPLE overhead
  - Trilinos’ sparse matrix-vector multiply uses 2-sided
  - Cost: message queue locking for MPI 2-sided (send, recv)
  - Vendors recommended switching to MPI 1-sided (MPI_Win), since optimized implementations don’t use MPI message queues
  - Trilinos has plans to explore this, but not this year

Conclusions

Sandia National Laboratories

- We want to deploy pipelined & communication-avoiding Krylov methods in Trilinos
  - Implementations exist now
  - We will put them in Trilinos this year
- Software challenges, because we want it to work for production users, instead of just hacking it in there
- We addressed some Belos & MPI – related challenges
- We need a better approach to asynchronous progress for nonblocking MPI operations

Thank you!!

Thank you!!

Sandia National Laboratories

- ECP PEEKS: This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, in support of the nations exascale computing imperative.