SAND2017-0711C



Summary of current thread parallelization efforts in Trilinos' linear algebra & solvers

Mark Hoemmen
Sandia National Laboratories
Tue, 07 Feb 2017

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. SAND2017-XXXXX C

Outline

- Tpetra's hardware & feature requirements
- Brief history of Tpetra
- Status: Thread-parallelizing solvers & preconditioners
  - I may need to stop here
  - But I include the rest as reference material
- Status: Support for multiple memory & execution spaces
- Status: Thread-parallel fill

What is Tpetra?

- Tpetra, a Trilinos package, implements
  - Sparse graphs, (block) sparse matrices, & dense (multi)vectors
  - Parallel kernels for solving Ax=b & Ax=λx
  - MPI communication & (re)distribution
  - Fill: {Create,modify} {graph,matrix,vector}
- Key Tpetra features
  - Can solve problems w/ > 10^9 unknowns
  - Can pick the type of values (real, complex, automatic differentiation, ensemble, …)
  - MPI + X (threads) parallelism

Must support > 3 architectures

Sandia National Laboratories

- Systems here (or nearly)
  - Trinity (Intel Haswell & KNL)
  - Sierra (CORAL): NVIDIA GPUs + IBM multicore CPUs
  - Clusters, workstations, etc.
- 3 different architectures
  - Multicore CPUs (big cores)
  - Manycore CPUs (small cores)
  - NVIDIA GPUs
- MPI only, & MPI + threads
  - Threads don't always pay on common CPU architectures
  - Don't slow down MPI-only case in legacy codes

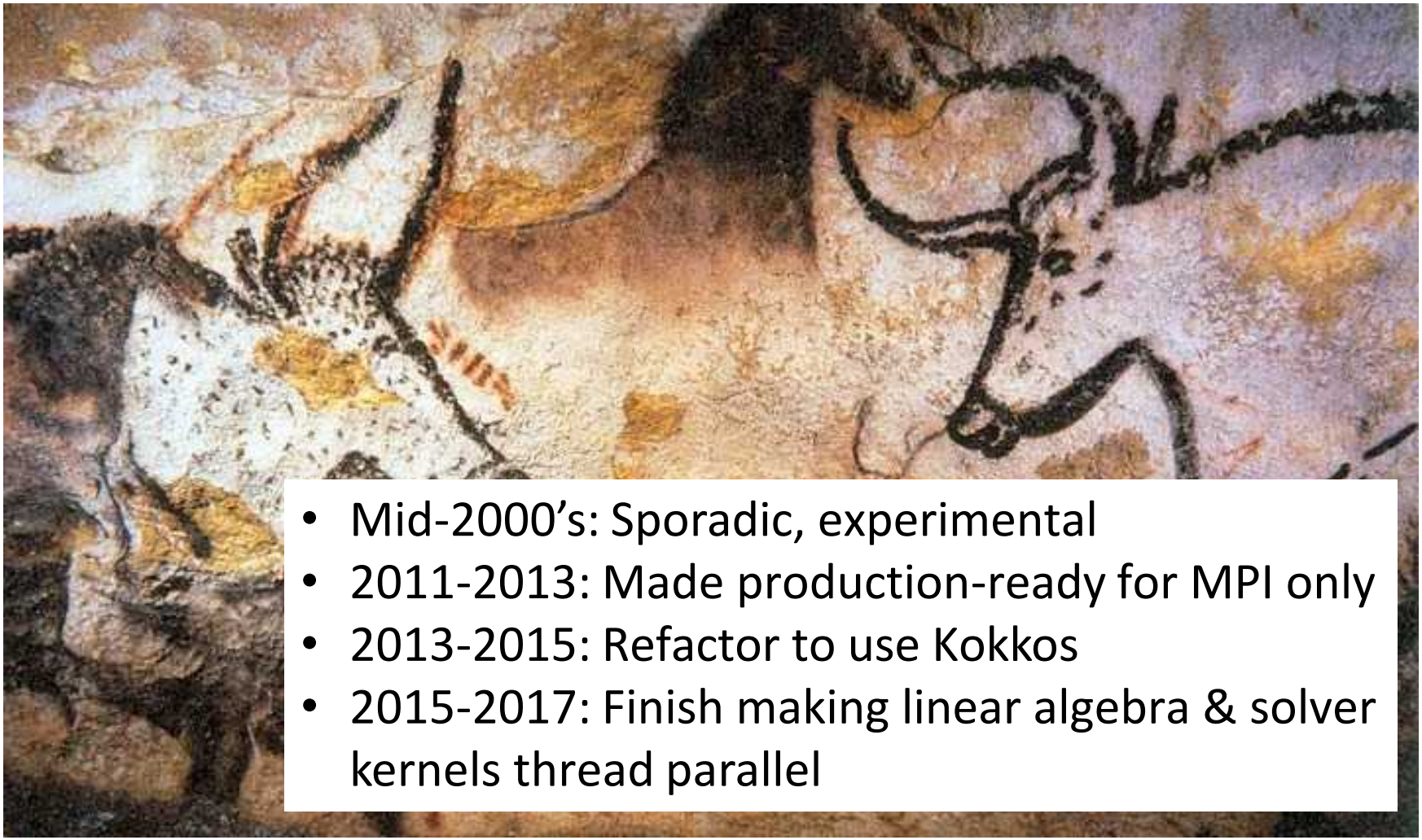
CORAL COLLABORATION
OAK RIDGE • ARGONNE • LIVERMORE

SIERRA

Kokkos as hedge against…

- Hardware divergence
- Parallel programming model
  - OpenMP, OpenACC, CUDA, TBB, Pthreads, Qthreads, …
- Traditional shared memory
  - vs. PGAS / distributed shared
  - e.g., MPI 1-sided
- Threads at all
  - Kokkos' semantics require vectorizable (ivdep) loops
- Kokkos protects our HUGE time investment of porting Tpetra

History of Tpetra



- Mid-2000's: Sporadic, experimental
- 2011-2013: Made production-ready for MPI only
- 2013-2015: Refactor to use Kokkos
- 2015-2017: Finish making linear algebra & solver kernels thread parallel

Status of solver-related kernels

Sandia National Laboratories

Categories of completeness

Sandia National Laboratories

- Thread-parallel kernel (TPK) exists
  - Our Kokkos kernel, &/or third-party libraries
  - Runs in parallel, correctly, on GPU
- Optimized (OPT)
  - Publications, brief measurements, or “not worrisome”
  - Multicore CPUs, KNL, & GPUs
  - Efforts underway now to make this more systematic
- Deployed (DPL) in Trilinos solvers
  - Not just a prototype code or unattached “bare” kernel
  - Can call it through Amesos2, Belos, Ifpack2, MueLu, etc.
  - If deployed but not thread-parallel yet, still “DPL”

Kernels & solvers done or nearly so

Sandia National Laboratories

Kernel name	TPK	OPT	DPL	Notes
CRS sparse mat-vec	Y	Y	Y	
Dot, Norm, Axp(b)y (update)	Y	Y	Y	(Multi)Vector; for Krylov
GEMV (GMRES orth.)	Y	Y *	Y	Depends on opt'd BLAS
Chebyshev (setup & solve)	Y	Y	Y	
Jacobi (setup & solve)	Y	Y	Y	
Gauss-Seidel (setup & solve)	Y	Y	Y **	Builds by default in develop
Sparse triangular solve	Y	Y	N	HTS (OpenMP) in ShyLU now; cuSPARSE for CUDA
BCRS (block) sparse mat-vec	Y	N	Y	OPT in progress
BCRS (block) Jacobi (s. & s.)	Y	N	Y	OPT in progress

TPK: Thread-parallel kernel exists
OPT: Optimized (CPU & GPU)
DPL: Deployed in Trilinos' solvers

Kernels & solvers in progress

Sandia National Laboratories

Kernel name	TPK	OPT	DPL	Notes
Sparse matrix-matrix multiply	Y	Y	Y*	MueLu setup; off by default
Explicit sparse transpose	N	N	N	MueLu setup (maybe)
Sparse (un)pack for comm	N	N	N	MueLu setup
CRS fillComplete	Y/N	Y/N	Y	MueLu setup; some done
{Ex,Im}port setup	N	N	N	MueLu setup
Sparse Cholesky	Y	Y	N	In progress
Sparse LU	Y	Y	N	In progress
ILU(k)	Y	Y	N	In progress
Line smoothing	N	N	Y	Many tridiagonal solves
BCRS line smoothing	N	N	N	In progress
Domain decomp. setup	N	N	N	Computing overlap

Future work (hard stuff)

Sandia National Laboratories

Kernel / solver name	TPK	OPT	DPL	Notes
FastILU (SPAI ILU)	Y	Ys	N	Fine-grained parallel setup. Needs algorithm research.
Algebraic multigrid aggregation (part of setup)	N	N	N	Needs algorithm research & lots of software work.
{Ex,Im}port execution (MPI comm. from multiple threads)	N	N	N	Depends on MPI support. Rewrite for 1-sided?
Space for your kernels here!				

Status of support for multiple memory & execution spaces

>1 memory or execution spaces

- Our upcoming platforms
  - Trinity (KNL): 2 memory spaces (HBM, DDR4)
  - Sierra: 2 exec & mem spaces (GPUs + multicore CPUs)
- Common hardware features
  - 2 memory spaces: “fast & small” vs. “slow & big”
  - Can access any mem space from any exec space (NUMA)
  - Limited “fast” (<1GB/core)
- Not just Kokkos’ problem: Tpetra, solvers, &/or apps must get involved

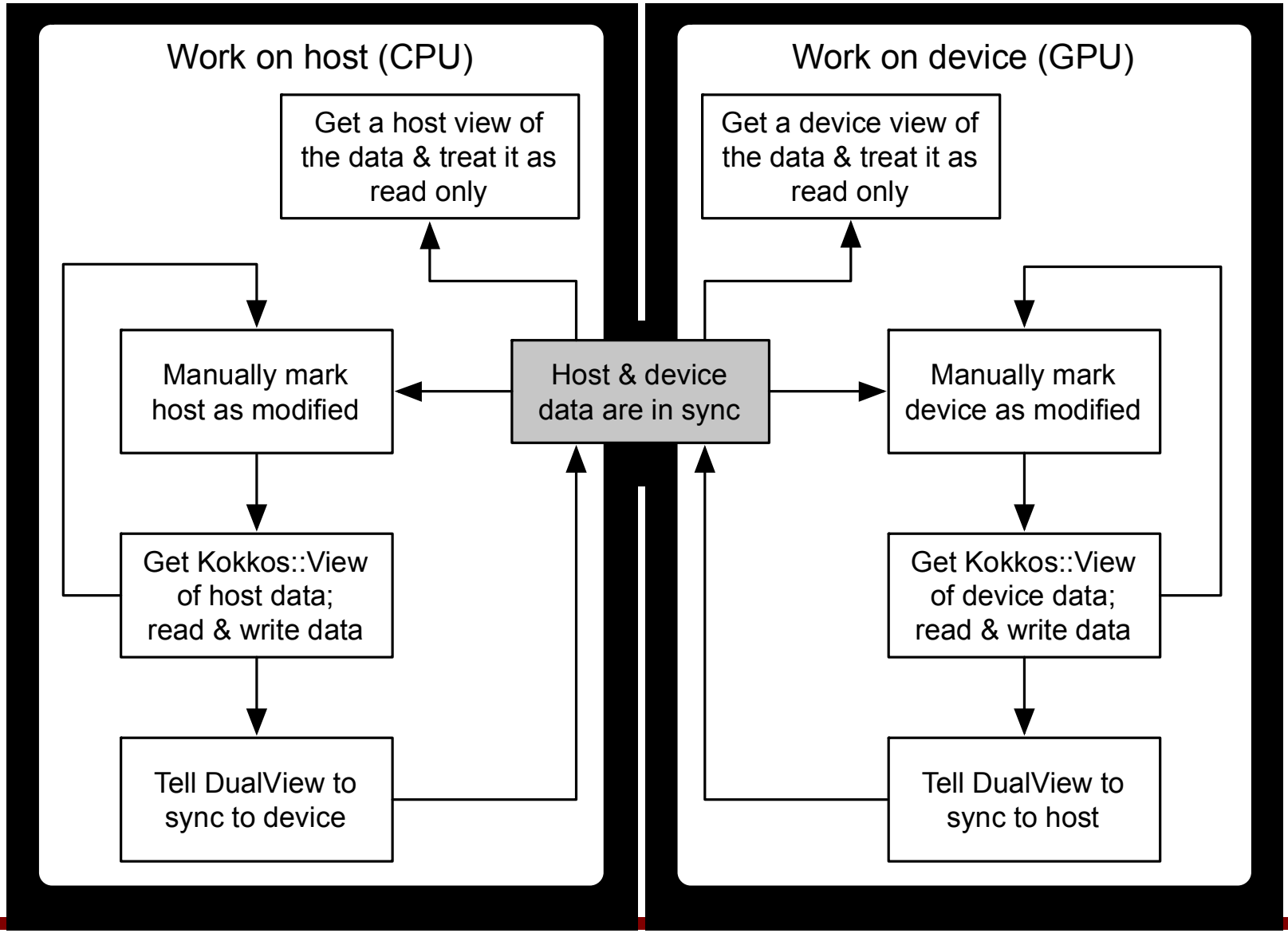
- Support 3 use cases
  1. Gradual porting (mix new Kokkos-ized code with legacy host code)
  2. Use “fast” memory as temp work space (page in / out as needed)
  3. Concurrently use 2 exec spaces (e.g., MPI pack on host, compute on device)

Strategies for 3 use cases

Sandia National Laboratories

- Dual view semantics: (1) & (2)
  - Handles gradual porting (1) & temp workspace (2) use cases
  - Successful use in LAMMPS (interactions btw user vs. GPU modules)
  - Only explicitly manages use of 1 execution space at a time
    - Tpetra *prefers* executing where most recent version of data live
    - Tpetra *may* execute in another space (e.g., overlap pack & compute)
- Concurrent execution in >1 exec spaces? (3)
  - Extension of how Kokkos handles data parallelism in tasks
  - Optional "execution space instance" argument to kernels
  - Exec space instance behaves like "stream" in CUDA
  - Would only expose capability; exploiting it is harder
    - Tpetra could overlap MPI-related (un)pack w/ host compute
    - Solvers & apps could overlap different kinds of work

Dual view semantics



Example of DualView use

- Tpetra::Vector<…, Cuda> X (map);
- { // Some code that works on CUDA
  - X.sync<Cuda> (); // copy changes to CUDA device only if needed
  - X.modify<Cuda> (); // we're changing in CUDA space
  - auto X_lcl = X.getLocalView<Cuda> ();
  - parallel_for (X.getLocalLength (), [=] (const LO i) {X_lcl(i) = …});
- } // done w/ CUDA. Don't sync unless will change on host.
- { // Some code that works (only) on host
  - X.sync<Host> (); // copy changes to host memory only if needed
  - X.modify<Host> (); // we're changing in host space
  - auto X_lcl = X.getLocalView<Host> ();
  - parallel_for (X.getLocalLength (), [=] (const LO i) {X_lcl(i) = …});
- } // done changing on host. Skip sync & modify if only 1 space.

Status of dual view semantics

Sandia National Laboratories

- Tpetra & downstream packages currently assume UVM
  - HostMirror::memory_space == memory_space == CudaUVMSpace
    - Haven't yet built Tpetra & downstream w/ 2 different memory spaces
    - Need to fix for using Tpetra for explicit HBM management on KNL
  - A little Tpetra & most downstream code assume host access
    - Forces CUDA_LAUNCH_BLOCKING=1, hindering task parallelism
    - sync() calls fence() ➔ correct use of dual view semantics would relax this
- Kokkos: CUDA ok, HBM (KNL) nearly ready
- MultiVector & Vector done (have dual view semantics)
- Next: BlockCrsMatrix , CrsMatrix, CrsGraph (last)
  - Dynamic graph structure changes may be host only for a while
  - Recommended: build Kokkos data structures & pass off to Tpetra

Status of thread-parallel fill

Sandia National Laboratories

Sparse linear algebra use pattern

Sandia National Laboratories

- Fill: Create / modify matrix & vector data structures
  - As many ways to do this as there are applications
  - e.g., iterate over rows, entries, mesh points, elements (FEM), volumes (FVM), aggregates (AMG), …
- Setup for solve (e.g., build preconditioner)
- Solve linear system(s), eigenvalue problems, etc.
  - Coarse-grained computational kernels (e.g., sparse mat-vec)
- Repeat
  - e.g., nonlinear iteration, time stepping, parameter study
  - Petra Object Model & Trilinos solvers optimized for reuse, e.g., of
    - Data structures (graph, basis vectors, allocations) &/or
    - Communication patterns

Thread-parallel fill justifies refactor

Sandia National Laboratories

- Typical use pattern for sparse linear algebra
  - Fill into matrix & vector data structures
  - Setup for solve (e.g., build preconditioner)
  - Solve linear system(s)
  - Repeat
- Need thread-parallel fill because fill & setup not free
  - Lessons from 2011-2013 refactor for applications:
    - Some solves are cheap, so fill & preconditioner setup time matter
    - Most actual runs use few MPI processes, so node performance matters
  - Amdahl's Law: Even if solves take 90% of time with 1 thread, if you have 10 threads, now (sequential) fill & setup are 50%
  - Preconditioners create matrices, so they need fill too

Thread-parallel fill still primitive

Sandia National Laboratories

- Stage 1 plan: “If you want thread-parallel fill, get Kokkos widget out of Tpetra object, & fill into that”
- Advantages
  - Preserve Tpetra interface backwards compatibility
  - CUDA ok (Tpetra’s interface is properly host only)
  - Force apps to buy in (they need to rethink algs anyway)
- Disadvantages
  - Appears to defeat a major purpose of switching to Tpetra
  - “Kokkos widget” is yet another public interface to support
  - Tpetra already hard enough to use (e.g., ~10 LoC to Map)

Thread-safe Tpetra fill partly works

Sandia National Laboratories

- Done for CrsMatrix & (Multi)Vector, for methods that
  - Don't change graph structure (no "insert" yet)
  - Don't cause MPI communication (sumIntoGlobal for off-process rows)
- Return error code / success count; don't throw on error
- No more internal temp array dynamic allocation
- Don't touch Teuchos::RCP reference count
  - Don't call any method that returns RCP, like get*Map()
- sumInto, transform: Atomic update option
  - Default: Use atomic updates if not Serial
- sumInto, replace, transform: Take Kokkos::View / raw ptr
  - Avoids Teuchos::ArrayView debug mode reference count issues

Next steps

Sandia National Laboratories

Next steps

- Thread-parallelize all needed solver / preconditioner kernels
- Ensure correct parallelization & good performance of kernels
- Improve interfaces & user experience for thread-parallel fill
- Support concurrent execution in multiple execution spaces
- Think about the right way to do MPI_THREAD_MULTIPLE

Thanks!

- Kokkos refactor of Tpetra has been & is a HUGE effort
- Tpetra is only the beginning – lots of solver effort too
- Many Trilinos developers have contributed in some way