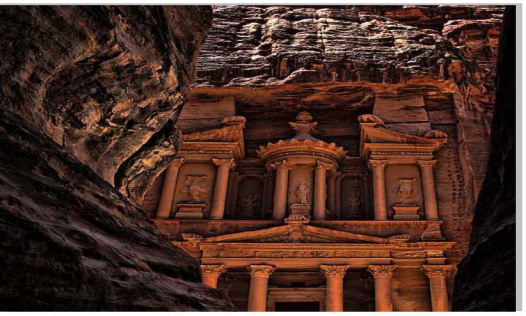
SAND2019-4556C



The latest in Tpetra: Trilinos' parallel sparse linear algebra

Mark Hoemmen
Sandia National Laboratories

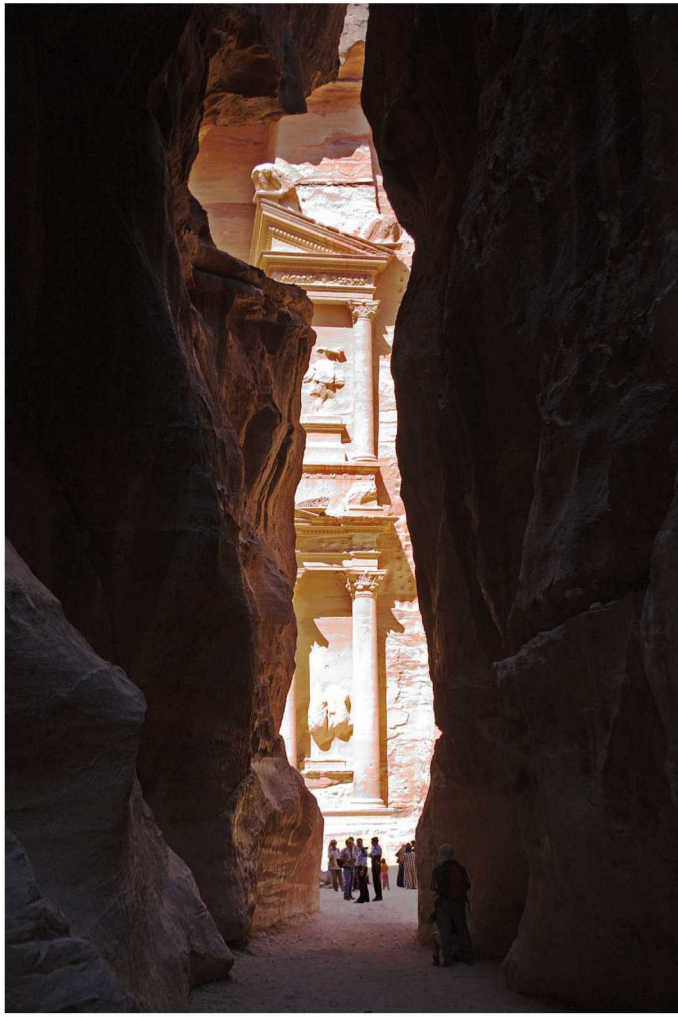
23 Apr 2019

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.

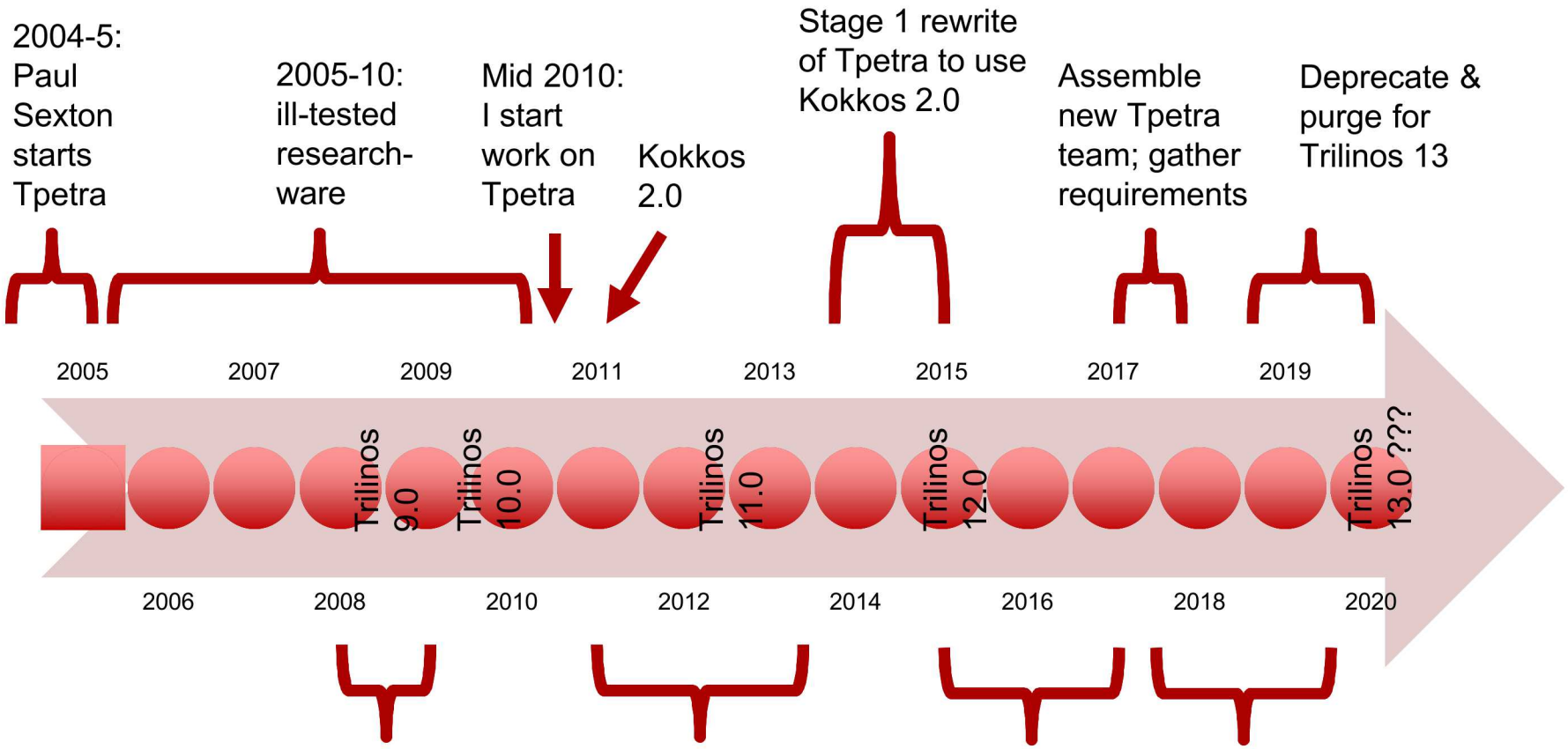
Tpetra: parallel sparse linear algebra

Sandia National Laboratories

- "Parallel": MPI + Kokkos
- Tpetra implements
  - Sparse graphs & matrices, & dense vecs
  - Parallel kernels for solving Ax=b & Ax=λx
  - MPI communication & (re)distribution
- Tpetra supports many linear solvers
- Key Tpetra features
  - Can manage > 2 billion (10^9) unknowns
  - Can pick the type of values:
    - Real, complex, extra precision
    - Automatic differentiation
    - Types for stochastic PDE discretizations



Over 15 years of Tpetra



Tpetra is a team now

Sandia National Laboratories

- Current team (2017-)
  - Karen Devine
  - Geoff Davidson
  - Tim Fuller
  - Mark Hoemmen
  - Jonathan Hu
  - Kyungjoo Kim
  - Chris Luchini
  - William Mclendon
  - Chris Siefert
- Plus contributions by
  - Victor Brunini
  - James Elliott
  - Christian Trott

How does Tpetra use Kokkos?

Sandia National Laboratories

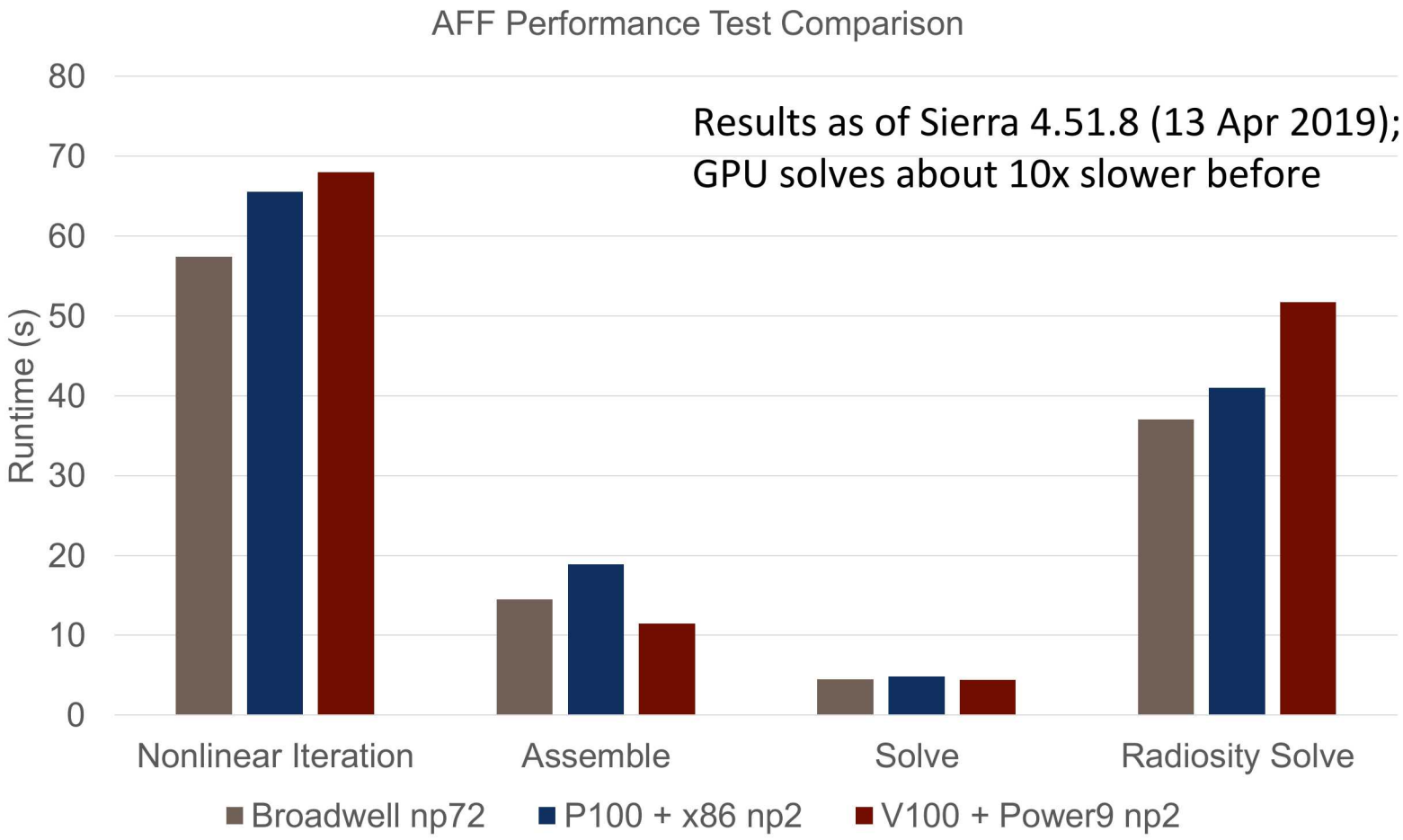
- Data structures
  - View: 1-3 D, scalars & integers, device & host
  - KokkosSparse::CrsMatrix (developed ~2013-4 for Tpetra)
  - DualView, StaticCrsGraph, UnorderedMap, UniqueToken
- Algorithms: sort, random
- Computational kernels (kokkos-kernels)
  - Dense BLAS & key sparse kernels
  - Kokkos-kernels started as part of Tpetra
- Many parallel loops (for, reduce, & scan)
  - Set up sparse data structures
  - Pack & unpack MPI communication buffers
  - “Long tail” of thread parallelization

Tpetra performance improvements

Sandia National Laboratories

- GPU + MPI performance
  - SPARC linear solver communication 7x faster on multiple GPUs
  - Aria AFF test ~10x faster on 2 GPUs than ~1 year before
- Highlights of some issues fixed
  - 4626: Avoid slow CUDA allocations in iterative linear solvers
    - Reuse allocation via static memory pool (eventually: write allocator?)
  - 4630: Make sure that local kernels use cuBLAS where possible
    - Aria AFF solve 2x faster overall; MultiVector X^T * Y time 23.7s to 1.2s
    - Made P100 np1 slightly faster than Broadwell np36 for full nonlinear iteration (76s vs. 84.5s)
  - 4417,4418: Avoid unneeded atomic updates in MultiVector unpack
    - Improved Tpetra solve times on multiple GPUs
    - Good idea for any parallel Kokkos back-end

Aria AFF performance



Lessons learned

Sandia National Laboratories

- CUDA, UVM, & host-pinned View allocations are SLOW
  - 100-20,000x (1 ms!!!) slower than HostSpace View allocations
  - (HostSpace only ~ 5x slower than raw new / delete)
  - ~30x slower than pairwise MPI_I{recv,send} exchanges on node
  - ➔ we already keep static allocations; considering memory pool
- Overexposing implementation hinders optimization
  - If users can get Kokkos::DualView out of MultiVector, then MultiVector can't lazily allocate device memory (see above)
- MPI: CUDA-aware != fast
- Most benefit: Careful measurement & tedious details
  - e.g., "Are you actually calling cuBLAS?"

Upcoming deprecations

Sandia National Laboratories

- LO, GO, & Node (3 of 4) template parameters
  - Simplify Trilinos' build logic
  - Improve build times & sizes
- DynamicProfile (in favor of StaticProfile)
  - {E,T}petra's sparse graphs & matrices had 3 options historically:
    - DynamicProfile: Expand storage per row on demand, unboundedly
    - StaticProfile: Strict upper bound on # entries / row, at constructor
    - Const graph: No structure changes post construction
  - We will remove DynamicProfile
  - Reduce code complexity (inherited from Epetra)
  - Remove impediments to thread parallelization
  - CrsGraph Export / Import will resize as needed

Tpetra's next steps

Sandia National Laboratories

- GPU performance improvements
  - Minimize device-host data movement
  - Minimize device & host-pinned memory allocations
  - Ensure correctness w/ CUDA_LAUNCH_BLOCKING off
- Improve asynchrony & latency hiding
  - Interface to let users exploit CUDA streams
  - Overlap MPI communication with other work
- Simplify access to Tpetra objects' local data
  - Easy parallel iteration over all (rows, entries, etc.)
  - Scope-based model of accessing local data

Kokkos wish list

Sandia National Laboratories

- Finish execution space instance (CUDA stream) support
  - Interfaces often exist in Kokkos, but ignore stream argument
- Improve build times & sizes
- Help w/ trade-offs: When is it worth…
  - … sync'ing data to device? (CUDA)
  - … launching parallel kernel? (CUDA, OpenMP)
  - Is this an abstraction on top of Kokkos?
- Team sort

Thanks!

Sandia National Laboratories

Extra slides

Sandia National Laboratories

Obstacles

Sandia National Laboratories

- Finding & making time to measure performance
  - Increasing pressure for solver features
  - Fragile build & multiplying configurations
- Trilinos & application build times