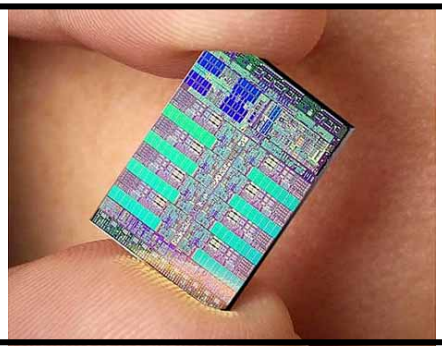
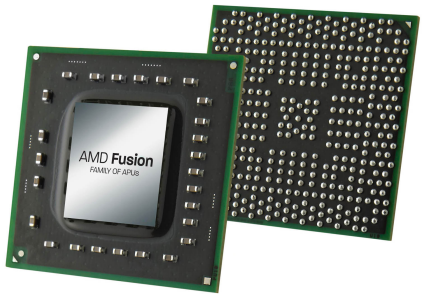
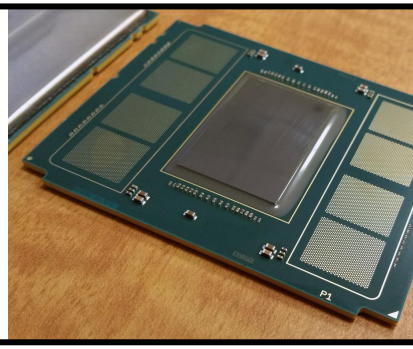
SAND2016-3623D

Exceptional service in the national interest

Sandia National Laboratories



Kokkos – Performance Portability Today

Christian Trott, Carter Edwards, Nathan Ellingwood, Si Hammond

crtrott@sandia.gov

Center for Computing Research

Sandia National Laboratories, NM

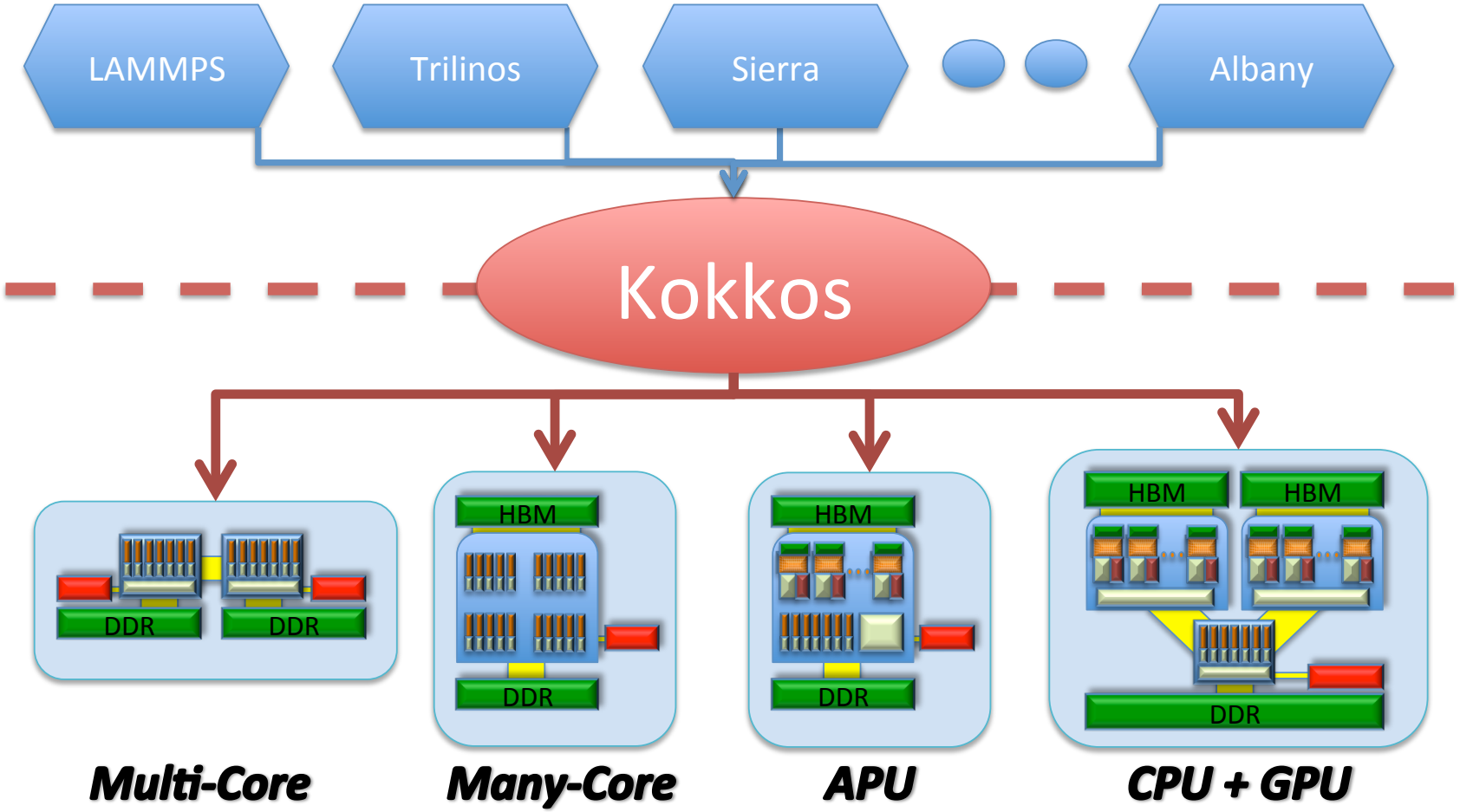
U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

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. SAND NO. 2016-2672 C

Kokkos: *Performance, Portability and Productivity*

Sandia National Laboratories

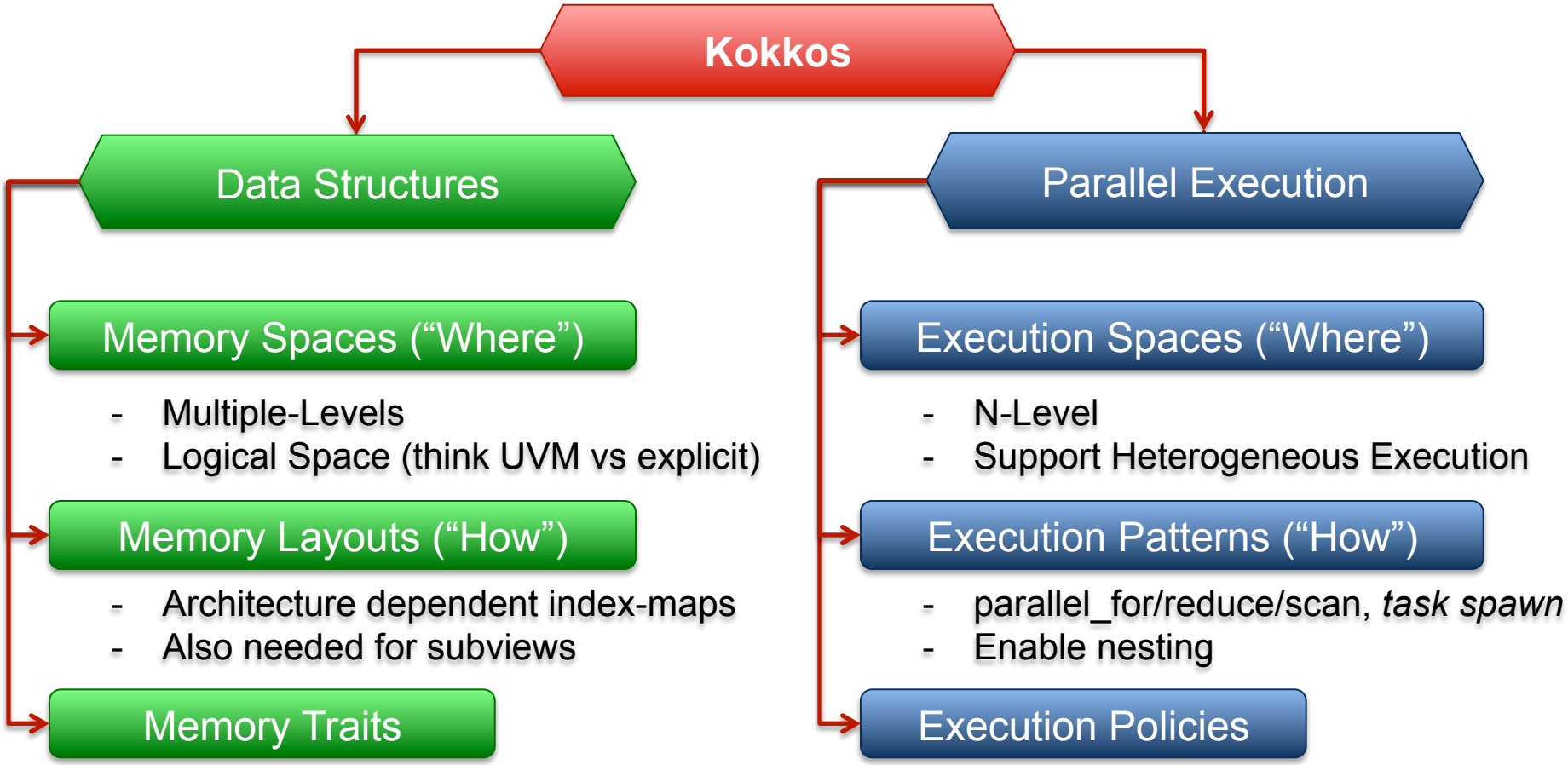


https://github.com/kokkos

Performance Portability through Abstraction

Sandia National Laboratories

Separating of Concerns for Future Systems…



Going Production

Sandia National Laboratories

- Robust Compilation and Environment Testing
  - Nightly test of 12 Compilers (GCC, Intel, Clang, NVCC)
  - >100 total configurations
  - Warnings as errors with "–pedantic -Wall –Wshadow ….."
- Documentation and User Training
  - Programming Guide
  - Extensive Tutorials: https://github.com/kokkos/kokkos-tutorials
    - > 300 Slides, dozens of hands-on examples with solutions
    - Under discussion: cloud based self-learning labs (used at GTC 2016)
- Profiling and Debugging Tools Integration
  - Talk by Simon Hammond (SNL) later this week
- Production and Next Generation Applications
  - ATDM targeting KNL and GPUs from beginning (talk by Stan Moore)
  - Sierra Mechanics focusing on thread safety/scalability until late 2017

Managing Memory Hierarchies

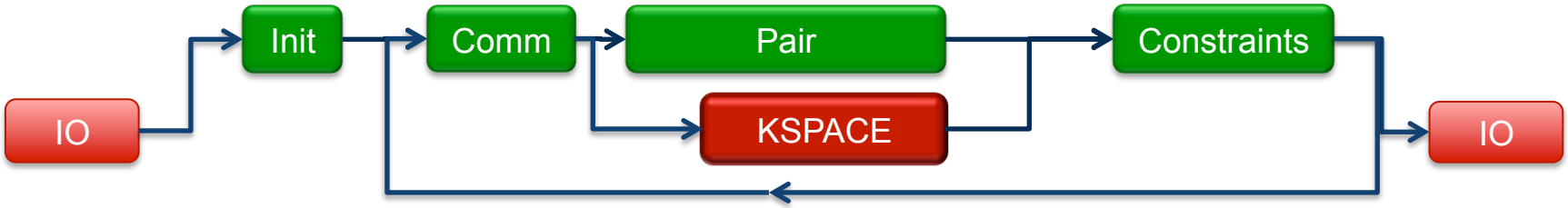
Sandia National Laboratories

- Memory Hierarchies At ***Low-Level*** are both Physical and Logical
  - Example current x86 + NVIDIA GPU with allocations in
    - UVM Space: let the CUDA runtime handle data transfer
    - CUDA Space: I know what I am doing, leave the runtime out of the way
    - HostPinned Space: I want to use asynchronous mem-copy with DMA engine
- Kokkos gives tools to do ***Low-Level*** management
- Applications write/use customizations for higher level management
  - Example LAMMPS:
    - Physics modules provide bit-masks for read/write access of fields
    - Memory management in LAMMPS uses Kokkos API to get data where it needs to be, including asynchronous copies
- Open: can we come up with generic ***High-Level*** Interfaces
  - Are use cases similar enough?
  - Multi-Lab CoE Talk by Ian Karlin (LLNL)

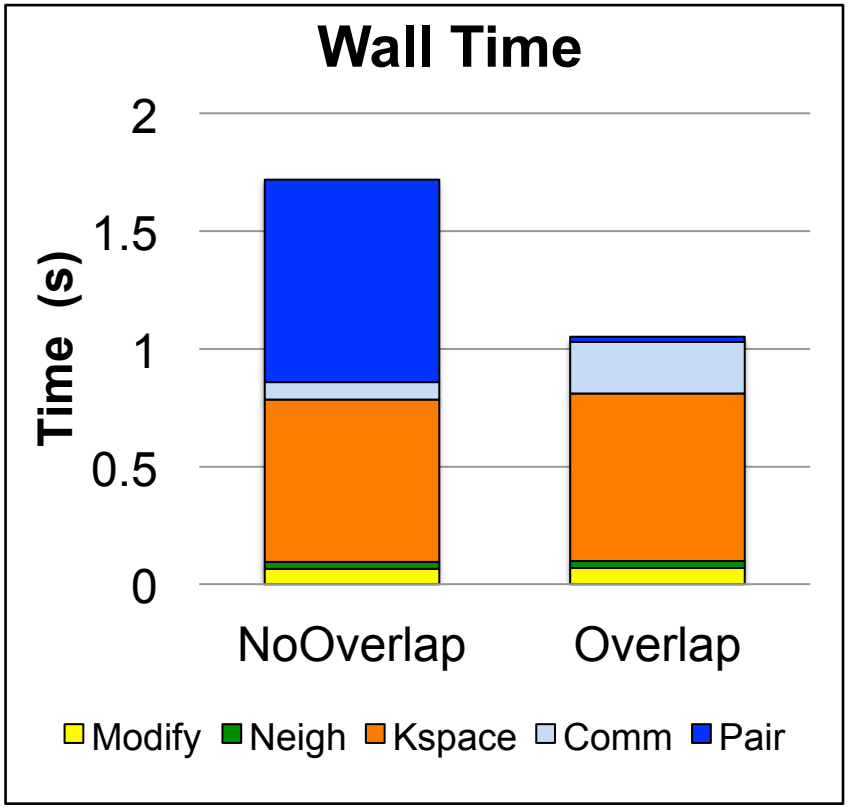
LAMMPS – Heterogeneous Execution

Sandia National Laboratories

Reverse Offload



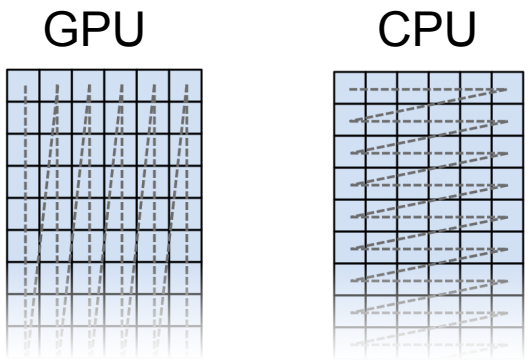
- LAMMPS/example/accelerate/in.phosphate
- Goal overlap Pair and Kspace
  - Requires Asynchronous Deep Copy
- When Overlapping:
  - Comm contains pair time since it fences to wait for pair force
  - 96% of Kspace time reduction

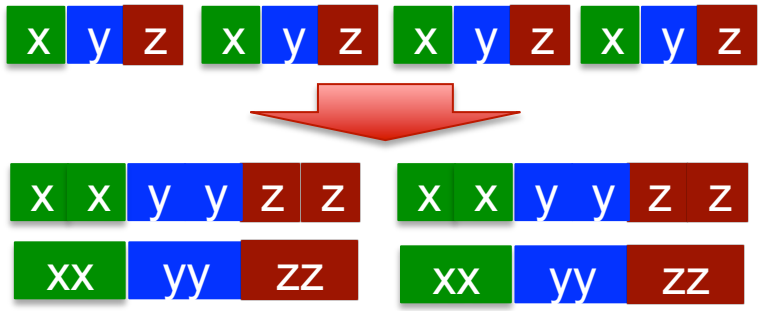


Managing Access Patterns

Sandia National Laboratories

- Change Data Access Pattern
  - Single typedef per code
- Adapt to Architectures
- Custom Layouts Easy
- Example SIMD friendly storage
  - Support explicit vector types for some kernels
  - `View<float*[3],LayoutRight>`
  - `View<float*[3],LayoutSIMDRight>`
  - `View<float2*[3],LayoutRight>`



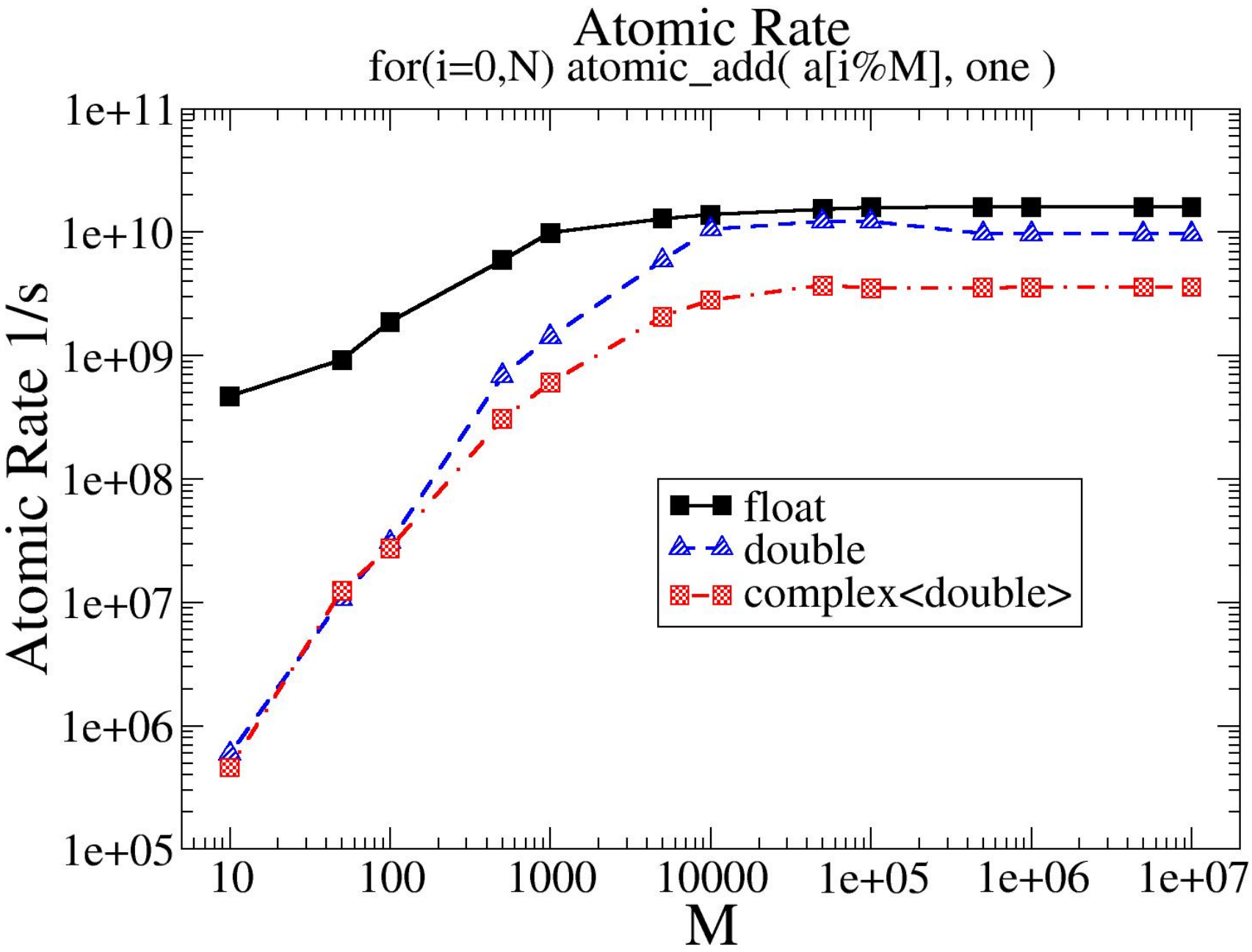


- Before: `a[i%V + j*V + (i/V)*V*3]` Now: `a(i,j)`

Example Atomic Support

Sandia National Laboratories

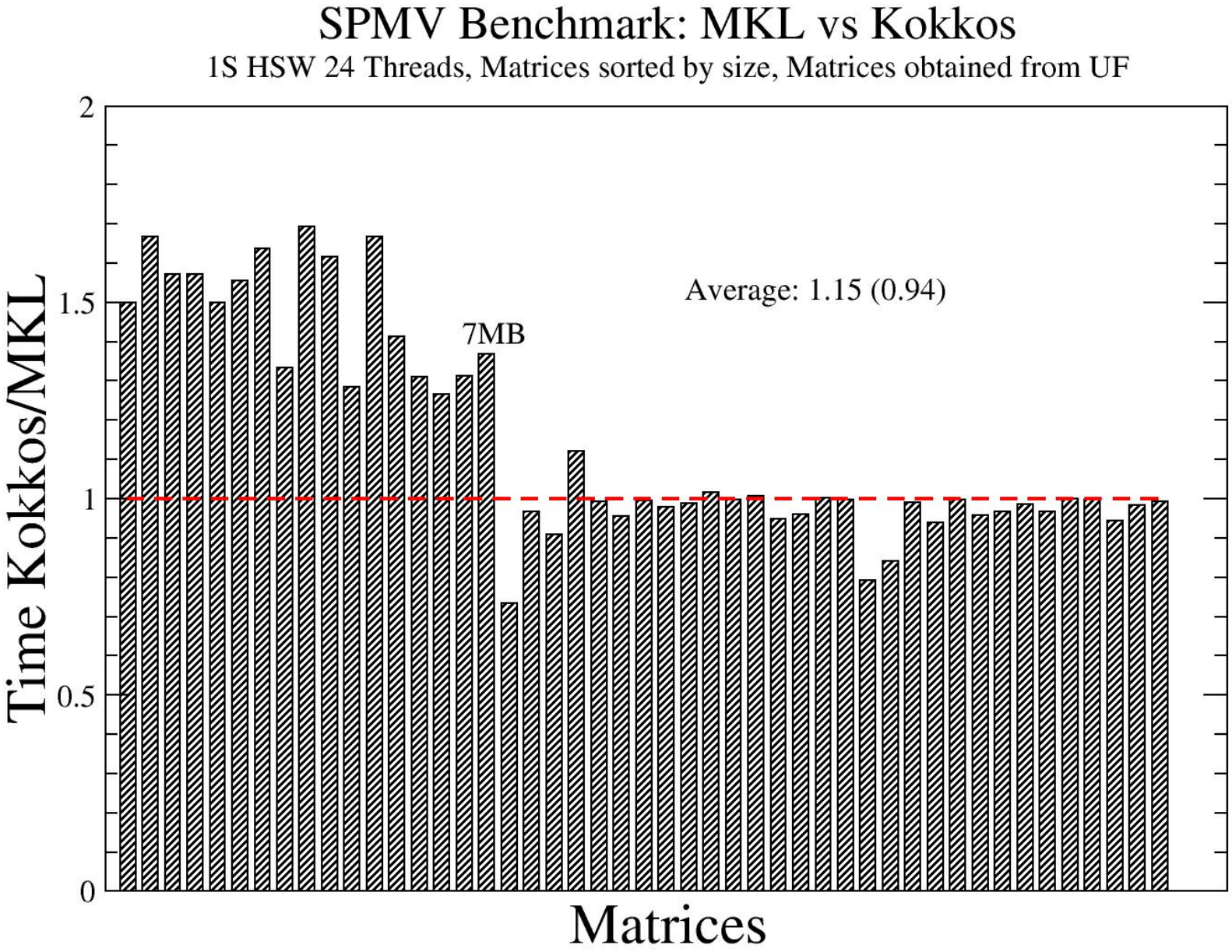
- Avoiding write conflicts comes with cost
  - ***Coloring:*** code complexity (for unstructured), potentially much more memory traffic (only parts of each cache line are used per color), loss of concurrency
  - ***Data Replication:*** memory footprint / traffic, additional reduction, less cache efficient
  - ***Atomics:*** serialization, loss of vectorization, potentially loss of L1 caching
  - ***Compute Replication:*** more flops / iops, more memory traffic
- Kokkos is used with all methods
- For ***unstructured*** problems atomics are often preferred over other approaches
  - `View<double**, MemoryTraits<Atomic> > a_atomic = a;`


Atomic Rate
for(i=0,N) atomic_add(a[i%M], one)
Atomic Rate 1/s
1e+11
1e+10
1e+09
1e+08
1e+07
1e+06
1e+05
10
100
1000
10000
1e+05
1e+06
1e+07
M
float
double
complex<double>

Under development: KokkosKernels

Sandia National Laboratories

- Provide BLAS (1,2,3); Sparse; Graph and Tensor Kernels
- No required dependencies other than Kokkos
- Local kernels (no MPI)
- Hooks in TPLs such as MKL or cuBLAS/cuSparse where applicable
- Provide kernels for all levels of hierarchical parallelism:
  - Global Kernels: use all execution resources available
  - Team Level Kernels: use a subset of threads for execution
  - Thread Level Kernels: utilize vectorization inside the kernel
  - Serial Kernels: provide elemental functions (OpenMP declare SIMD)
- Work started based on customer priorities; expect multi-year effort for broad coverage
- People: Many developers from Trilinos contribute
  - Consolidate node level reusable kernels previously distributed over multiple packages



On GPUs: CuSparse vs Kokkos: All: 1.07 Without Small: 0.84

The Common Problems We Face

Sandia National Laboratories

- Interference with Compiler Optimizations
  - Deducing Independence of Views: **restrict** for pointer as class members?
  - Hoisting loads from inner loops, with that being parallel_for
  - Loosing “const” when creating lambdas with capture by reference
  - Generally loosing surrounding information when using Lambdas
- Deficiencies in C++ Language for threading models
  - *this capture for Lambdas in member functions
    - Part of C++17
    - Enables asynchronous dispatch
    - Added to Clang 3.9
  - Error handling in threaded environments
  - OpenMP 4 handling of classes
  - What to do with STL objects

The Way Forward

Sandia National Laboratories

- Stabilize Kokkos Capabilities
  - Support tasking on all platforms
  - Make sure compilers optimize through layers
  - Harden KNL support for High Bandwidth Memory
- Broaden Implementation Coverage for Kokkos Kernels
- Support Production Teams in Adoption
- Develop even more Documentation
- Extend profiling tools to help with transition

www.github.com/kokkos/kokkos:	Kokkos Core Repository
www.github.com/kokkos/kokkos-tutorials:	Kokkos Tutorial Material
www.github.com/kokkos/kokkos-tools:	Kokkos Profiling Tools
www.github.com/trilinos/Trilinos:	Trilinos Repository

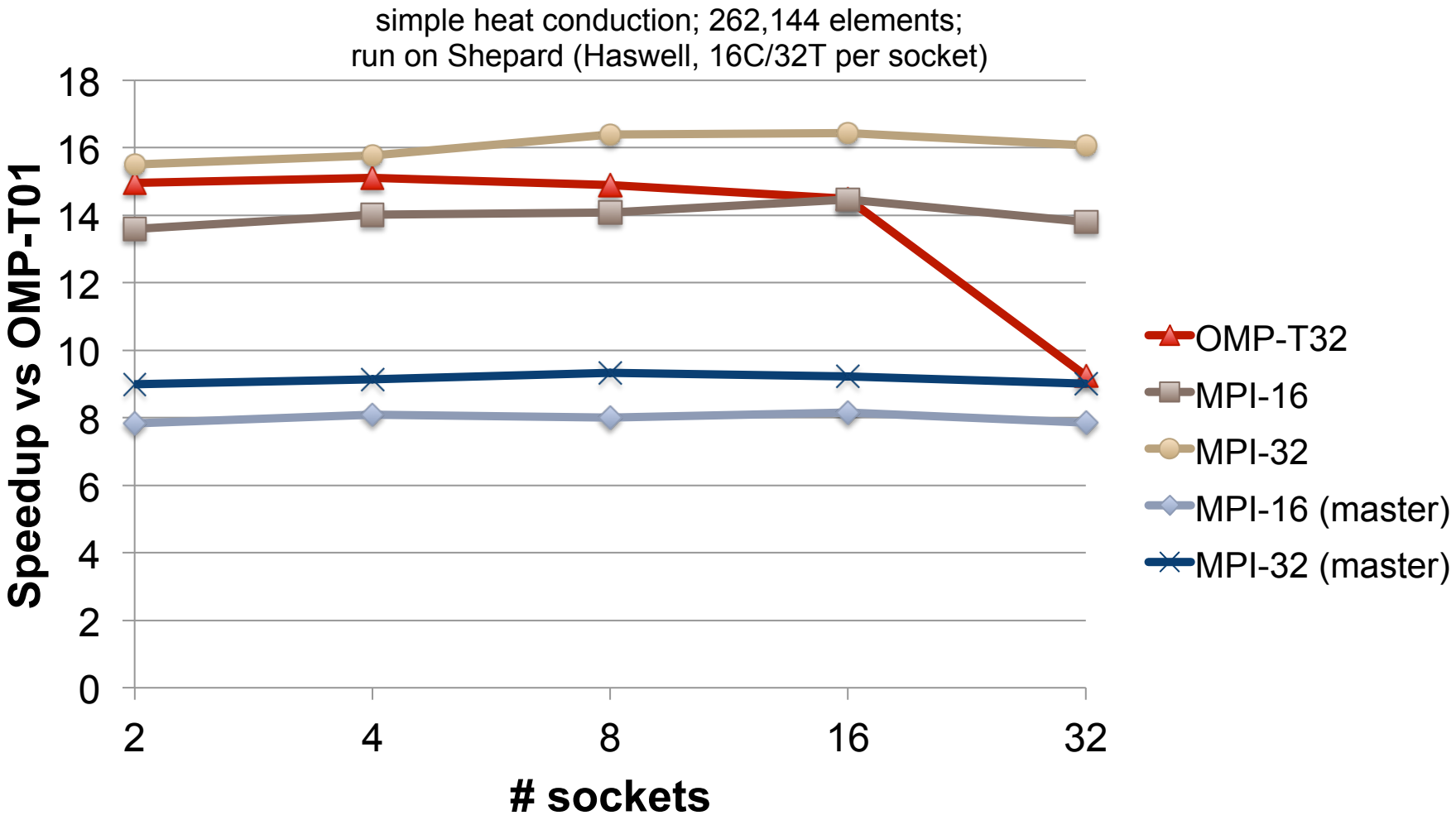
Sandia National Laboratories

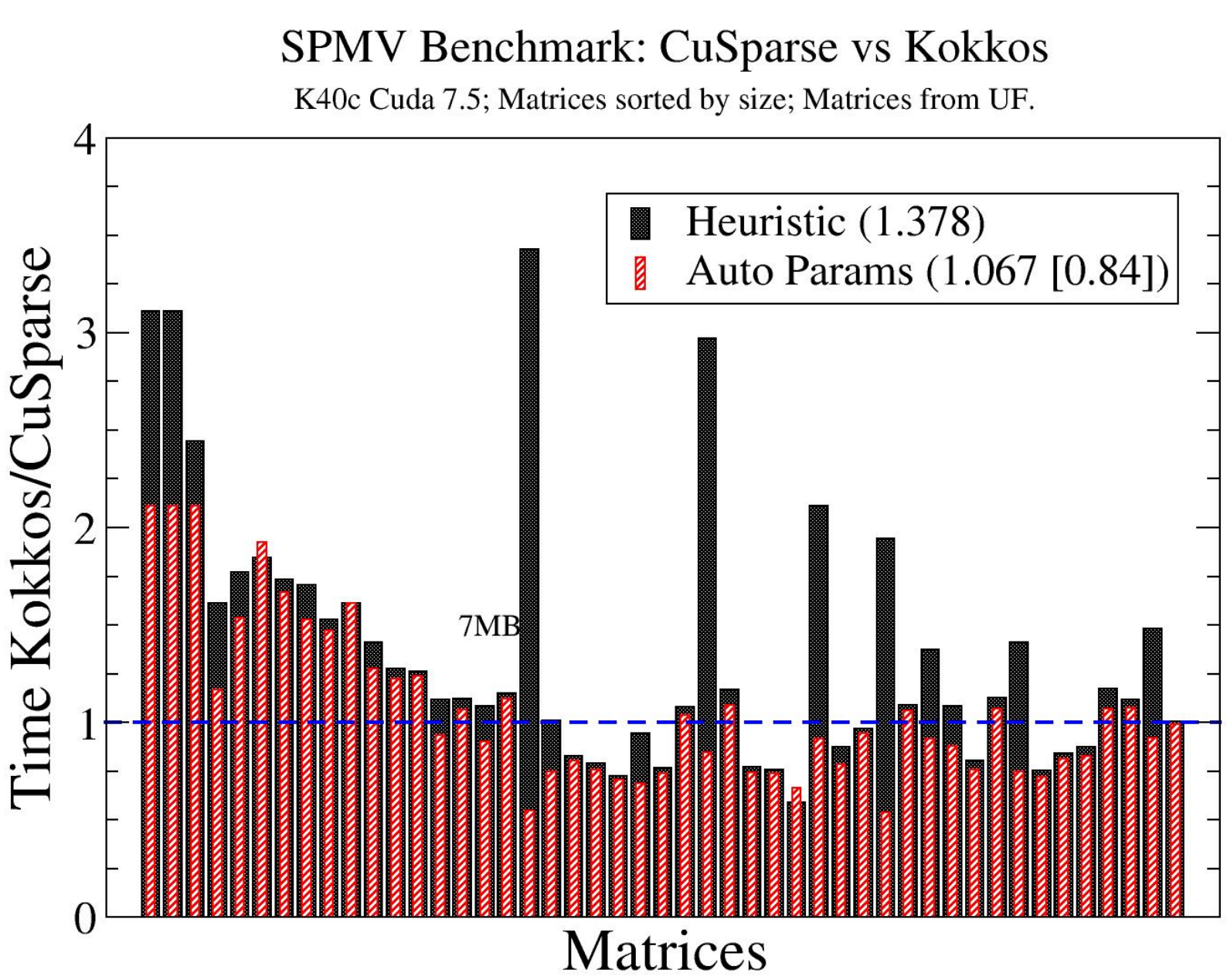
Exceptional service in the national interest

http://www.github.com/kokkos

NALU Assembly

- Uses atomic operations to fill into matrix




SPMV Benchmark: CuSparse vs Kokkos
K40c Cuda 7.5; Matrices sorted by size; Matrices from UF.
Heuristic (1.378)
Auto Params (1.067 [0.84])
7MB
Time Kokkos/CuSparse
Matrices
0
1
2
3
4