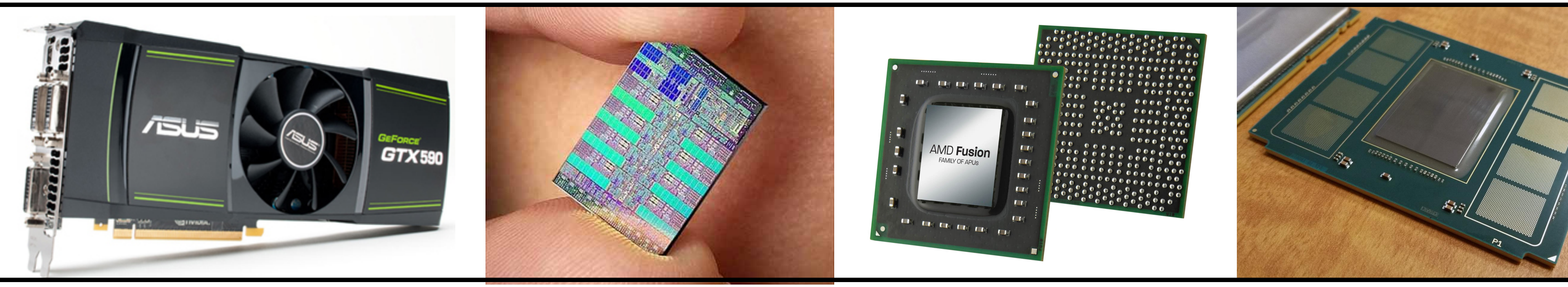
Exceptional service in the national interest

Sandia National Laboratories



Preparing Sandia's Application Portfolio for the Future Using Kokkos

Christian Trott, Daniel Sunderland, Carter Edwards, Si Hammond

crtrott@sandia.gov

Center for Computing Research

Sandia National Laboratories, NM

SAND2017-2110 C

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

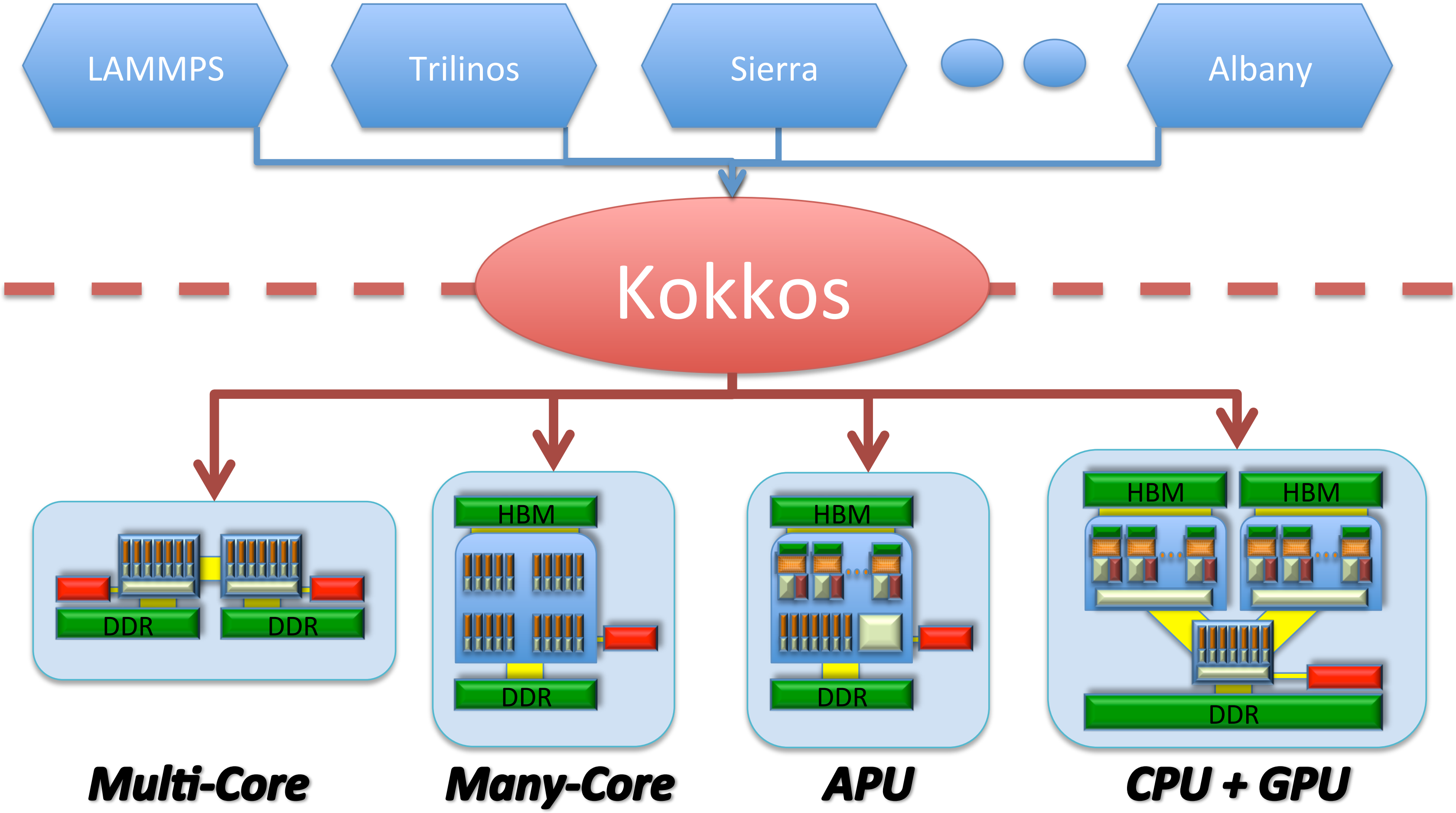
New Programming Models

- HPC is at a Crossroads
  - Diversifying Hardware Architectures
  - More parallelism necessitates paradigm shift from MPI-only
- Need for New Programming Models
  - Performance Portability: OpenMP 4.5, OpenACC, Kokkos, RAJA, SyCL, C++20?, …
  - Resilience and Load Balancing: Legion, HPX, UPC++, ...
- Vendor decoupling drives external development

My (slightly changed) Goal for the Talk:

Describe what it took to get Kokkos accepted by legacy applications

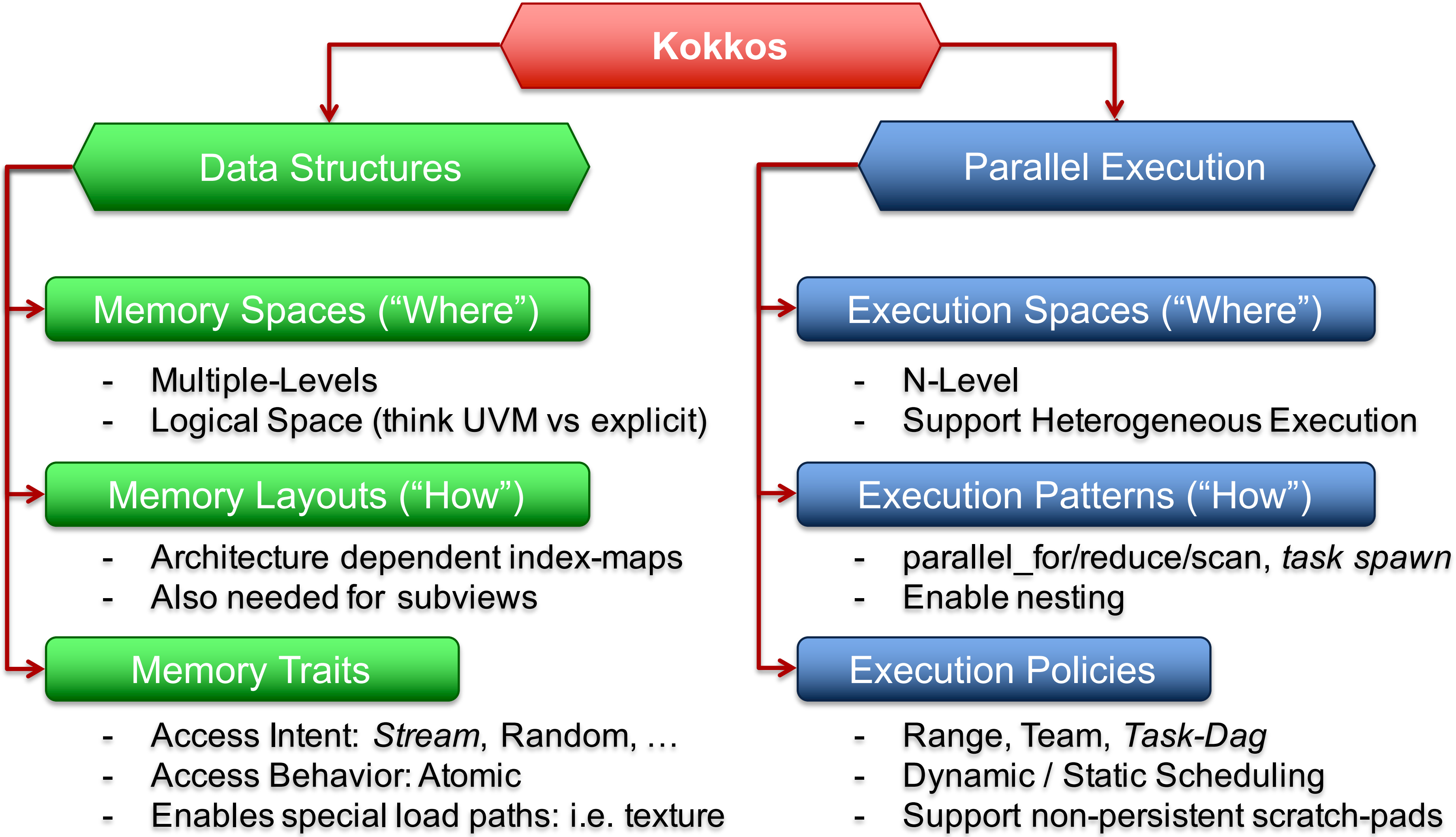
Kokkos: *Performance, Portability and Productivity*



https://github.com/kokkos

Performance Portability through Abstraction

Separating of Concerns for Future Systems…



Kokkos

Data Structures

Memory Spaces ("Where")

- Multiple-Levels
- Logical Space (think UVM vs explicit)

Memory Layouts ("How")

- Architecture dependent index-maps
- Also needed for subviews

Memory Traits

- Access Intent: *Stream*, Random, …
- Access Behavior: Atomic
- Enables special load paths: i.e. texture

Parallel Execution

Execution Spaces ("Where")

- N-Level
- Support Heterogeneous Execution

Execution Patterns ("How")

- parallel_for/reduce/scan, *task spawn*
- Enable nesting

Execution Policies

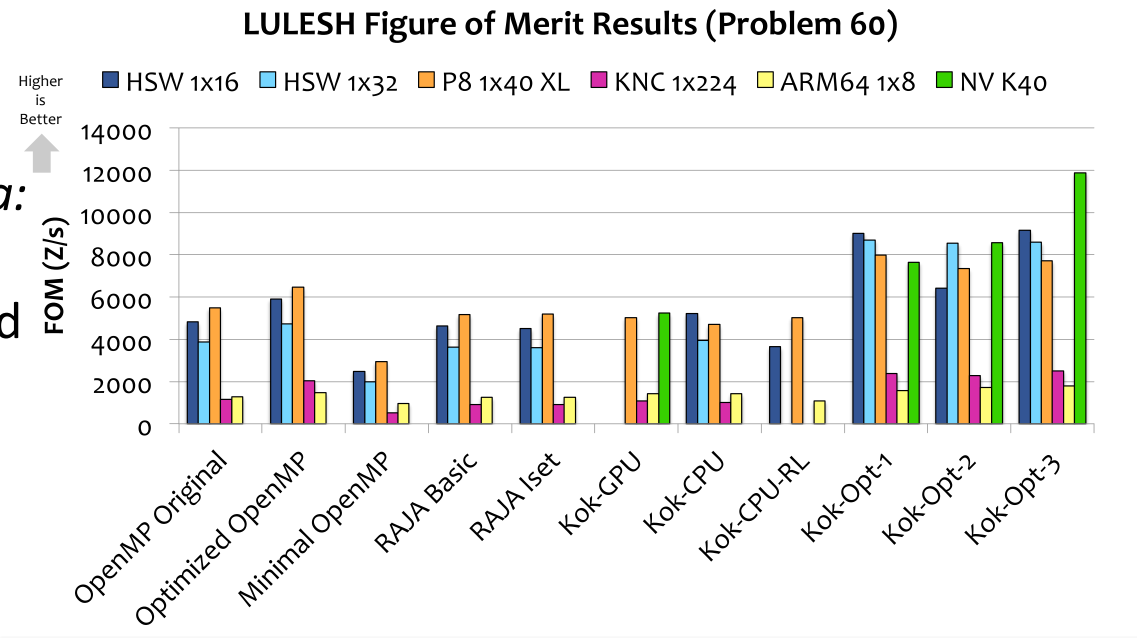
- Range, Team, *Task-Dag*
- Dynamic / Static Scheduling
- Support non-persistent scratch-pads

Timeline

2008	***Initial Kokkos:*** Linear Algebra for Trilinos
2011	***Restart of Kokkos:*** Scope now Programming Model
2012	***Mantevo MiniApps:*** Compare Kokkos to other Models
2013	***LAMMPS:*** Demonstrate Legacy App Transition
2014	***Trilinos:*** Move Tpetra over to use Kokkos Views
	Multiple Apps start exploring (Albany, Uintah, …)
2015	***Github Release of Kokkos 2.0***
2016	***Sandia Multiday Tutorial*** (~80 attendees)
	Sandia Decision to prefer Kokkos over other models
2017	***DOE Exascale Computing Project*** starts
	Kokkos-Kernels and ***Kokkos-Tools*** Release

Initial Demonstrations

- Demonstrate Feasibility of Performance Portability
  - Development of a number of MiniApps from different science domains
- Demonstrate Low Performance Loss versus Native Models
  - MiniApps are implemented in various programming models
- DOE TriLab Collaboration
  - Show Kokkos works for other labs app
  - *Note this is historical data:* Improvements were found, RAJA implemented similar optimization etc.

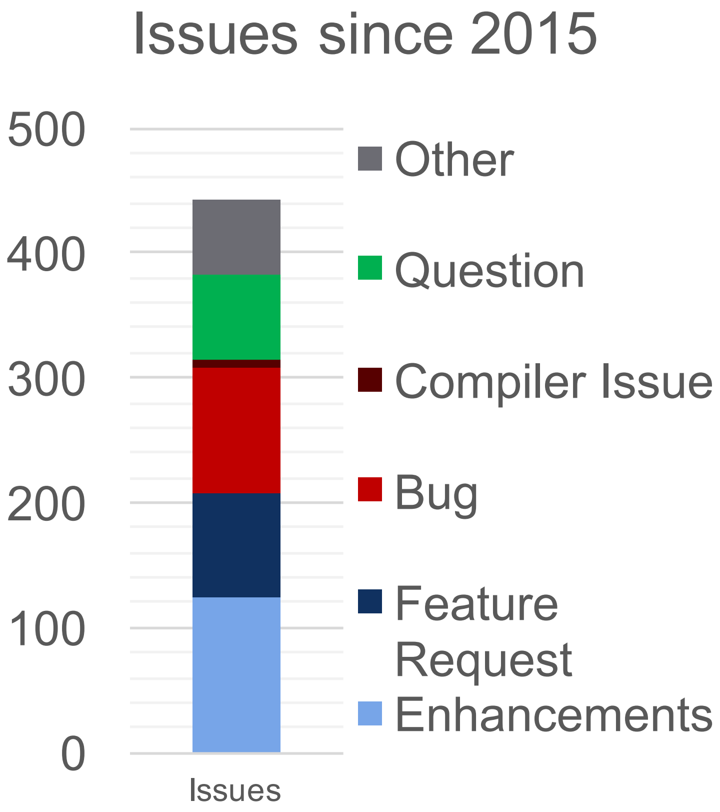


Training the User-Base

- Typical Legacy Application Developer
  - Science Background
  - Mostly Serial Coding (MPI apps usually have communication layer few people touch)
  - Little hardware background, little parallel programming experience
- Not sufficient to teach Programming Model Syntax
  - Need training in parallel programming techniques
  - Teach fundamental hardware knowledge (how does CPU, MIC and GPU differ, and what does it mean for my code)
  - Need training in performance profiling
- Regular Kokkos Tutorials
  - ~200 slides, 9 hands-on exercises to teach parallel programming techniques, performance considerations and Kokkos
  - Held at GTC, and SC; Also at request of institutions
  - Now dedicated ECP Kokkos support project: develop online support community

Keeping Applications Happy

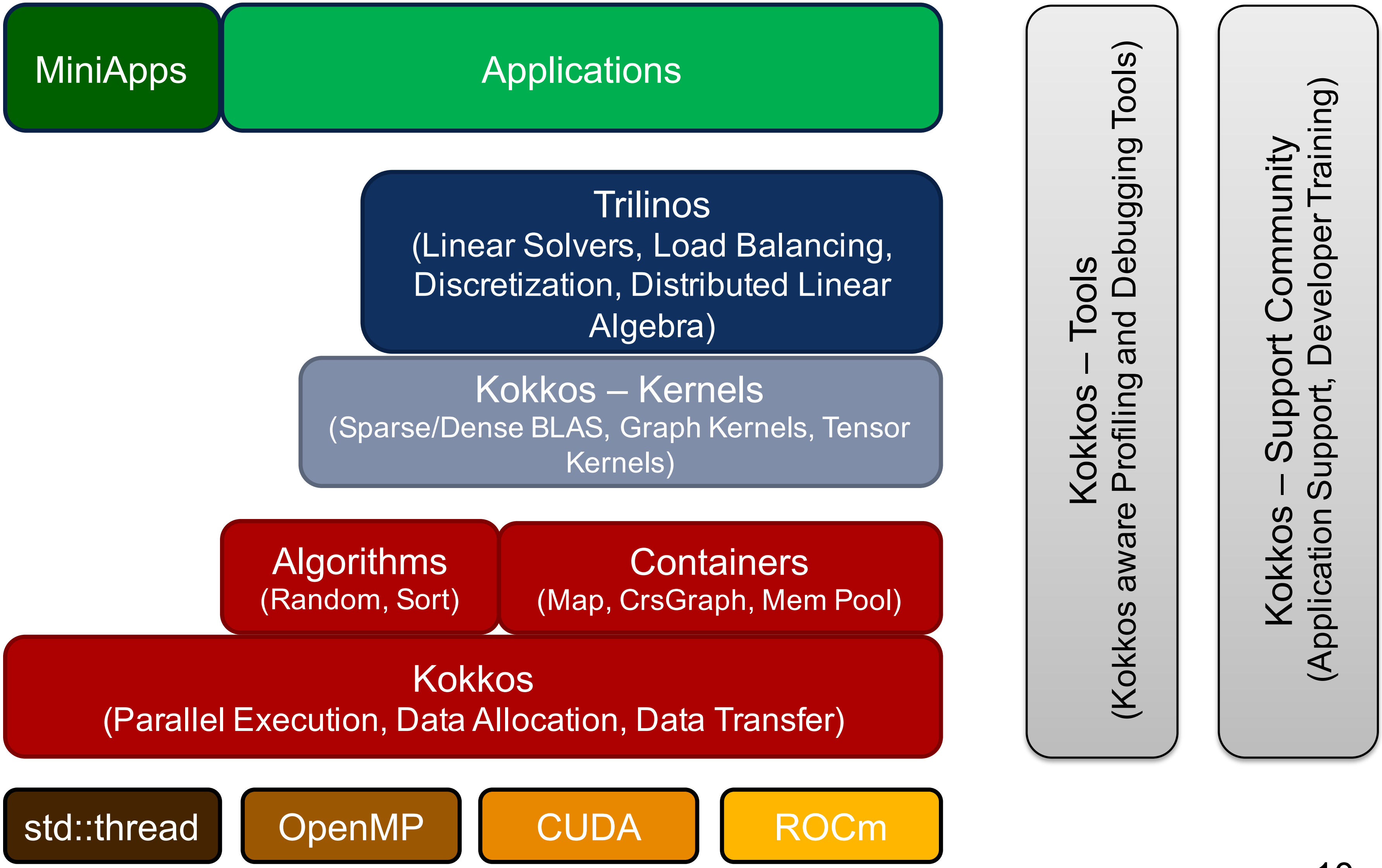
- Never underestimate developers ability to find new corner cases!!
  - Having a Programming Model deployed in MiniApps or a single big app is very different from having half a dozen multi-million line code customers.
  - 430 Issues in 22 months
  - ~25% are small enhancements
  - ~20% bigger feature requests
  - ~25% are bugs: often corner cases
- Example: Subviews
  - Initially data type needed to match including compile time dimensions
  - Allow compile/runtime conversion
  - Allow Layout conversion if possible
  - Automatically find best layout
  - Add subview patterns



Testing and Software Quality

- Programming Models are invasive
  - Reach many code locations: all parallelizable loops
  - Some take over low level data structures
  - Potentially costly to back out again
- Performance Portability implies multi platform
  - Much greater variety of compilers and architectures
  - Programming model needs to support union of customer needs
- Testing on SNL Testbeds
  - Intel Haswell, KNL; IBM Power; Cavium ARM; NVIDIA Kepler, Pascal
  - 15 compilers (GCC, Intel, Clang, IBM, PGI)
  - >200 configurations every night
- SEMS: Support Common Software Stack accross SNL
  - Application teams don't have the resources for multiple software stacks
  - Deliver tested compiler/tpl combinations across diverse machines

Building an EcoSystem



MiniApps

Applications

Trilinos
(Linear Solvers, Load Balancing, Discretization, Distributed Linear Algebra)

Kokkos – Kernels
(Sparse/Dense BLAS, Graph Kernels, Tensor Kernels)

Algorithms
(Random, Sort)

Containers
(Map, CrsGraph, Mem Pool)

Kokkos
(Parallel Execution, Data Allocation, Data Transfer)

std::thread

OpenMP

CUDA

ROCm

Kokkos – Tools
(Kokkos aware Profiling and Debugging Tools)

Kokkos – Support Community
(Application Support, Developer Training)

Necessary Resources

- Long term development:
  - ~6 years effort so far
  - only now seriously working on major applications
- Now more Resources for Support/Tools than core Model R&D
  - ~ 2 FTE on core Kokkos development
  - ~ 1.5 FTE application support
  - ~ 2 FTE on Tools and Kokkos Kernels
- Diverse hardware resources for testing and development
  - Equivalent of 2-3 nodes for dedicated testing
  - ~5 different architecture testbeds for development
  - Beta access to all major HPC compilers
- Intensive Collaboration with Vendors
  - Working on Compiler Bugs, Compiler improvements and new backends

Further Material

- https://github.com/kokkos Kokkos Github Organization
  - **Kokkos:** *Core library, Containers, Algorithms*
  - **Kokkos-Kernels:** *Sparse and Dense BLAS, Graph, Tensor (under development)*
  - **Kokkos-Tools:** *Profiling and Debugging*
  - **Kokkos-MiniApps:** *MiniApp repository and links*
  - **Kokkos-Tutorials:** *Extensive Tutorials with Hands-On Exercises*
- https://cs.sandia.gov Publications (search for ’Kokkos’)
  - Many Presentations on Kokkos and its use in libraries and apps
- www.gputechconf.com/gtcnew/on-demand-gtc.php
  - Search for Kokkos: recorded talks on Kokkos and some usage

Sandia
National
Laboratories

Exceptional service in the national interest

http://www.github.com/kokkos