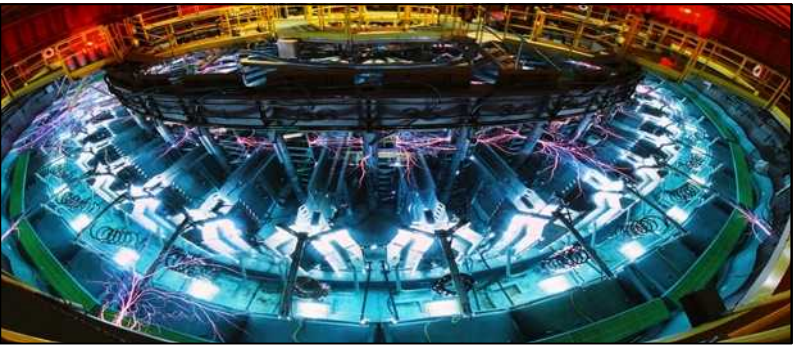
SAND2016-6233C

Exceptional service in the national interest

Sandia National Laboratories



Balancing Productivity, Portability and Performance - The Challenge for Programming Models at Exascale?

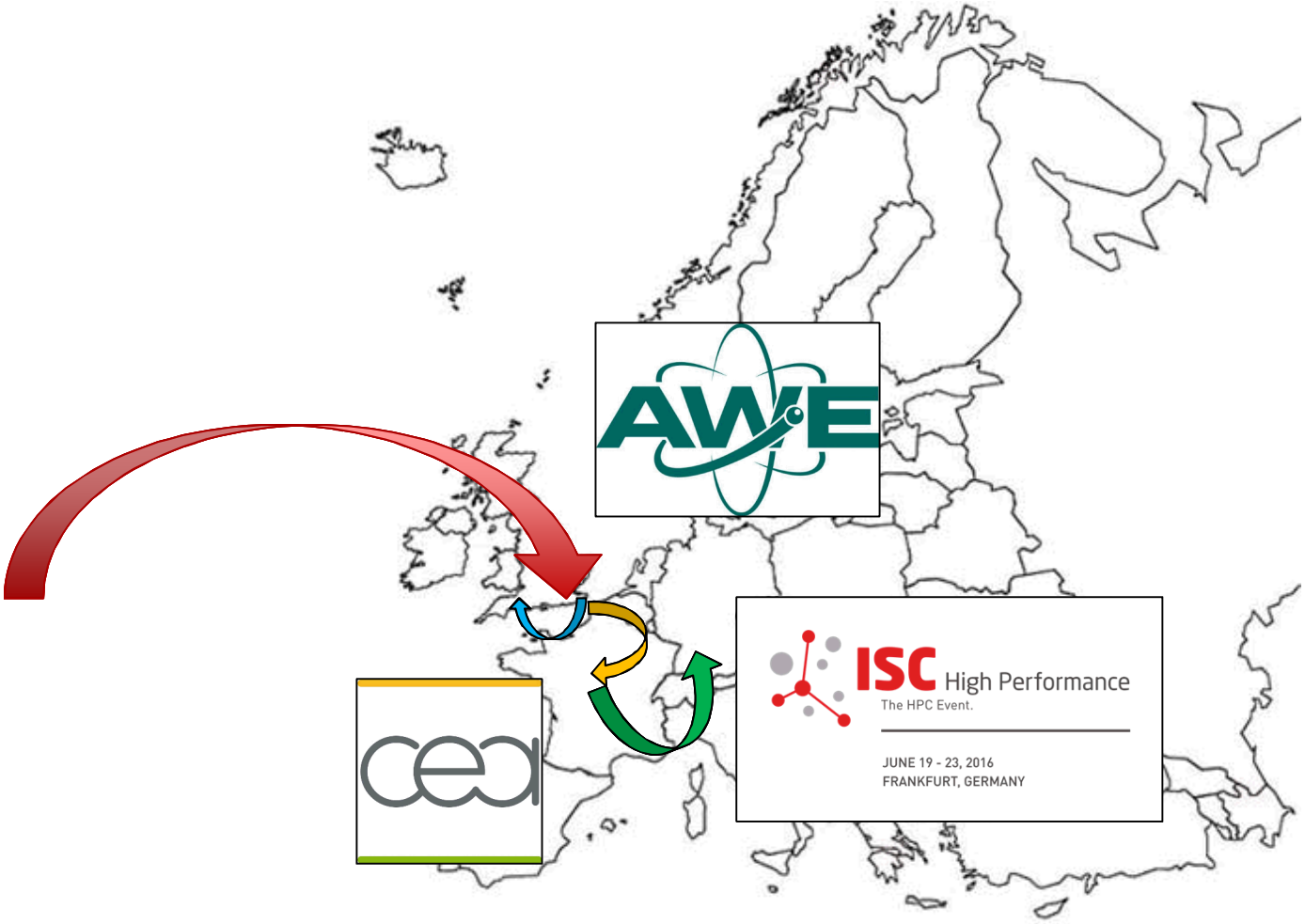
Scalable Computer Architectures
Sandia National Laboratories, NM
sdhammo@sandia.gov

U.S. DEPARTMENT OF ENERGY
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.

It’s Been a Long Three Weeks…

Sandia National Laboratories



Common Theme in Discussions…

Sandia National Laboratories

- Worrying less about the hardware…
  - Vendors are showing some great projections and ideas for the future
  - Incredible amount of innovation still left on the runway
  - Yes, this needs funding and we're working on it

- Much more worried about the software stack and applications…
  - Complex balance in pushing technology, maintaining legacy and still giving us a viable path to the forward
  - No clear alignment with the analytics community who seem to like Java, Python and scripting (this does not fly at the labs!)

- **"Credible Exa* machines" aren't credible until they solve real science**

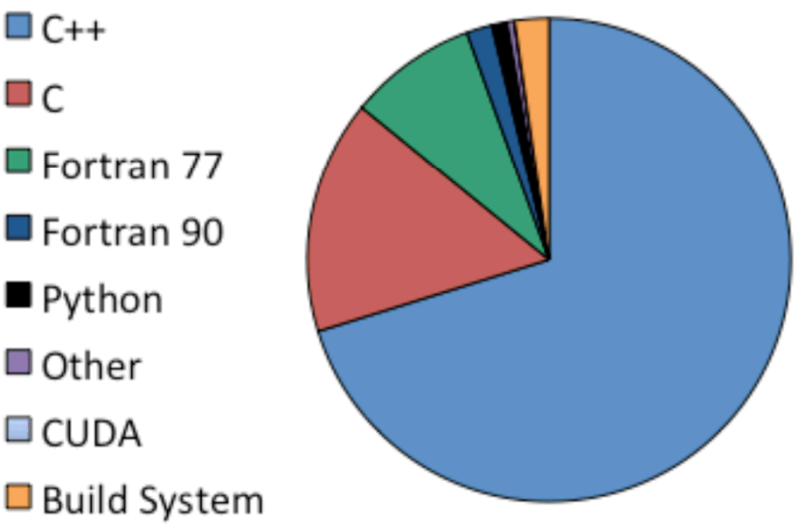
So What Are We Talking About Here?

Sandia National Laboratories

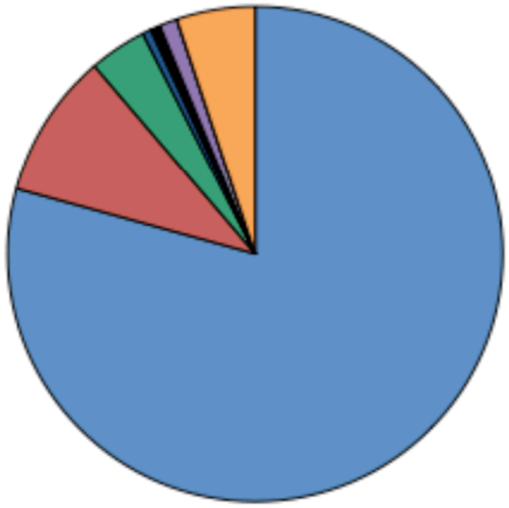
Represents ~10 applications out of >100 we have to run

Used throughout our code base and the community

Several Sandia Engineering Applications



Sandia Mathematics / Solvers TPL

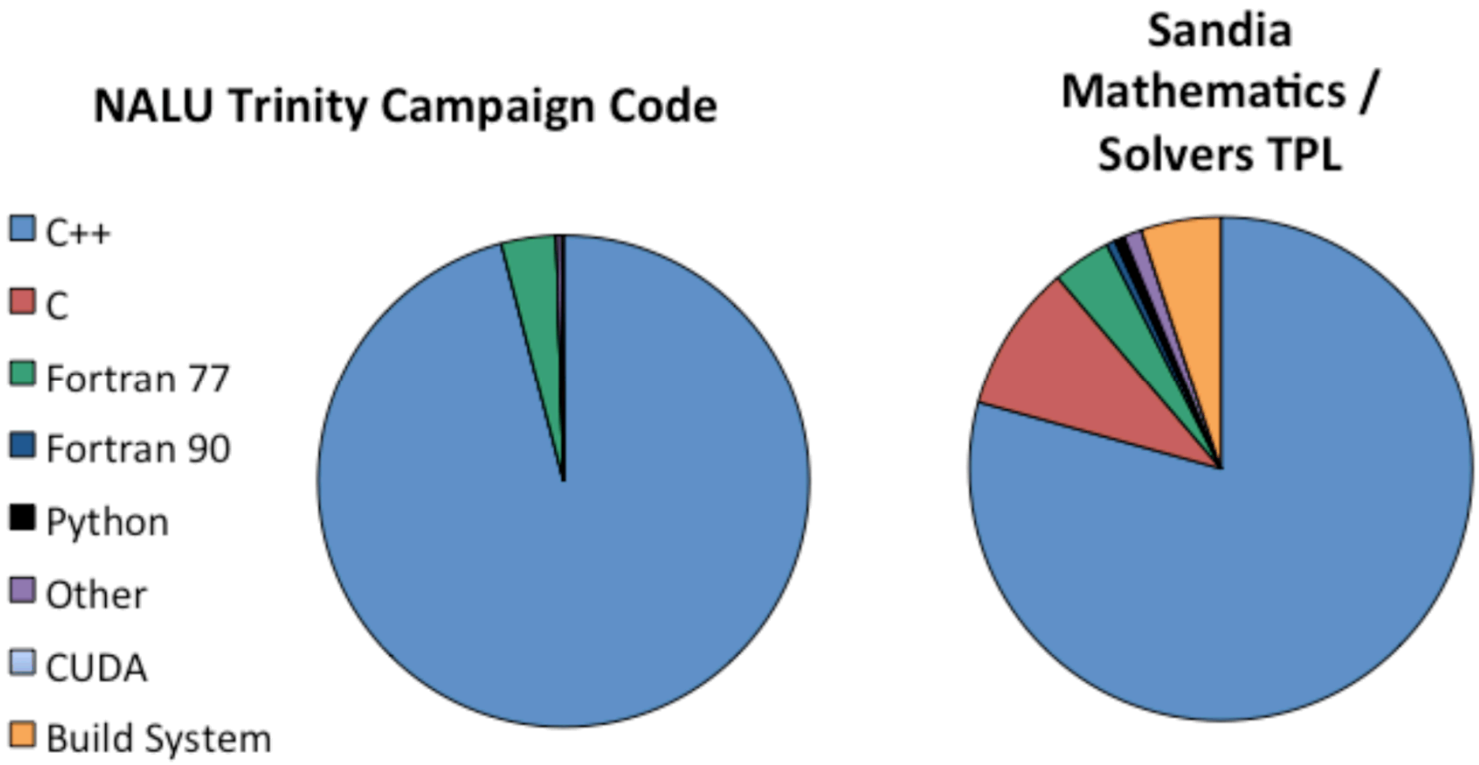


~11M SLOC of Application Code
(>50 Third Party Libraries **not** included)

~5M SLOC of Library Code
(>20 Third Party Libraries **not** included)

So What Are We Talking About Here?

Sandia National Laboratories



~65K SLOC of Application Code
(~5 Third Party Libraries **not** included)

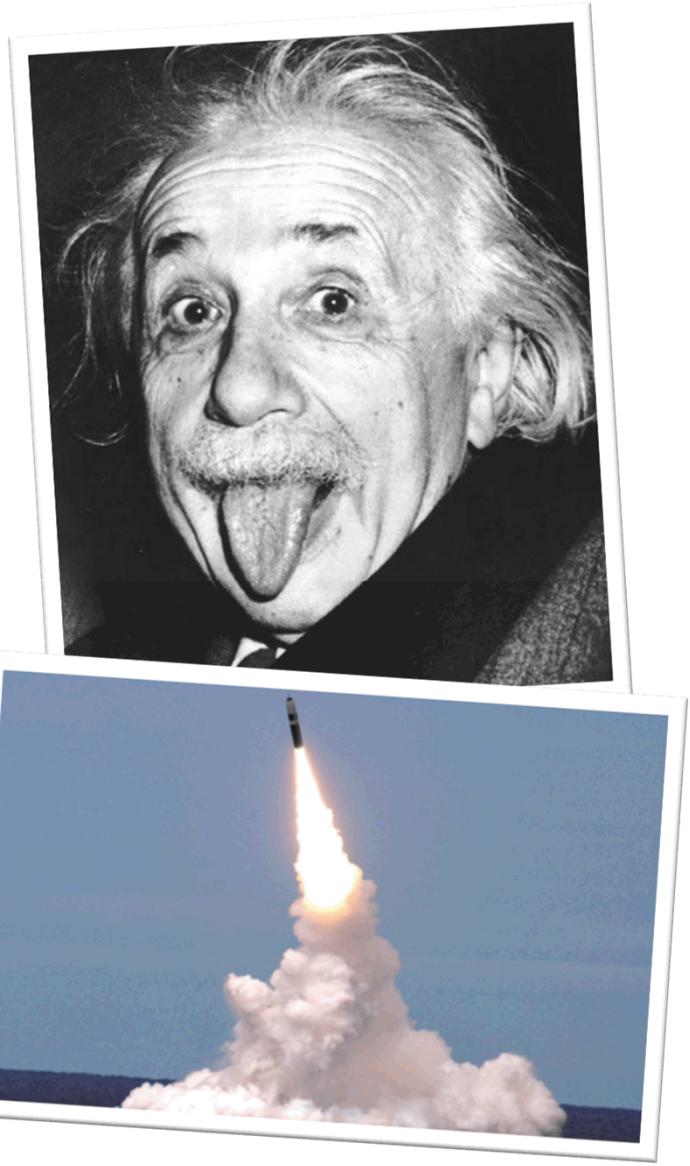
~5M SLOC of Library Code
(>20 Third Party Libraries **not** included)

http://www.trilinos.org

And there's more …

- **The complexity of this code is extremely high**
  - Written by a team of specialists who are very often the leading computational experts in an engineering or physics field
  - We don't want to document all of our algorithms and numerical techniques in literature (so developers often encounter them first time at the labs)

- **The code is well trusted/validated**
  - Our codes are often developed to qualify complex situations where physical testing can be difficult
  - Codes are verified and validated over a long period costing huge amounts of time and money



We do very cool things too..

Sandia National Laboratories

Sandia's self-guided bullet prototype can hit target a mile away

402

ALBUQUERQUE, N.M — Take two Sandia National Laboratories engineers who are hunters, get them talking about the sport and it shouldn't be surprising when the conversation leads to a patented design for a self-guided bullet that could help war fighters. (Click here for a video showing the prototype's flight.)

Sandia researchers Red Jones and Brian Kast and their colleagues have invented a dart-like, self-guided bullet for small-caliber, smooth-bore firearms that could hit laser-designated targets at distances of more than a mile (about 2,000 meters).

"We have a very promising technology to guide small projectiles that could be fully developed inexpensively and rapidly," Jones said.

Sandia is seeking a private company partner to complete testing of the

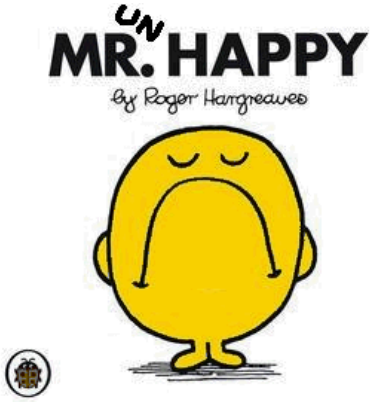
We do very cool things too…

Sandia National Laboratories

“Wow, you guys are screwed..”

- **Personal opinion is no, we are making *huge* progress but it is hard (and quite slow)**

- Major worry is that we will have Exascale hardware without a good software/science story



It is *this* community that can change these problems

Sandia National Laboratories

So What Are We Doing…?

Architecture Test Beds

Sandia National Laboratories

- **Intel:**
  - Sandy Bridge, Ivy Bridge and Haswell
  - Knights Corner & Knights Landing
- **AMD:**
  - Llano and Kaveri APUs
- **IBM:**
  - POWER7+ and POWER8
- **NVIDIA:**
  - K20, K40M and K80
- **ARM:**
  - APM X-Gene 1 and Cavium
- **Cray:**
  - Aries/Ivy Bridge

- Typically 30 – 50 node with Infiniband
- Few other one/two node test blades

Mantevo Mini-Apps

Sandia National Laboratories

- Mini-applications are condensed (but representative) versions of our application codes
  - Typically O(10,000) lines of code
  - Minimize third party libraries (easy to build)
  - Hopefully simplified build systems
- Incredibly popular with vendors and academia
  - Get representative kernels which focuses research and impact
- Available from ASCR Codesign Centers, ASC Codesign and Mantevo
- http://www.mantevo.org

Sandia National Laboratories

Performance, Portability And Productivity…

Programming Models and our Code Future…

Let's Start at the Beginning…

Sandia National Laboratories

- **Today almost all of our code is MPI everywhere**
  - No threading
  - Very little vectorization work
  - Not well optimized

What's Next…

Sandia National Laboratories

```
const Scalar*  xcoefs = &x.coefs[0];
const Scalar*  ycoefs = &y.coefs[0];
MINIFE_SCALAR  result = 0;

#pragma omp target teams map(xcoefs[0:n], ycoefs[0:n]) \
  map(tofrom: result) reduction(+:result)
{
    #pragma omp distribute parallel for schedule(static,1) reduction(+:result)
    for(int i=0; i<n; ++i) {
        result += xcoefs[i] * ycoefs[i];
    }
}
```

- Perhaps the most obvious choice is adding OpenMP
  - Turns out this is a lot of work
  - Isn't always performant
  - Still not widely supported (certainly not everything in the standard)

This is just intended to be representative of the problems in real codes

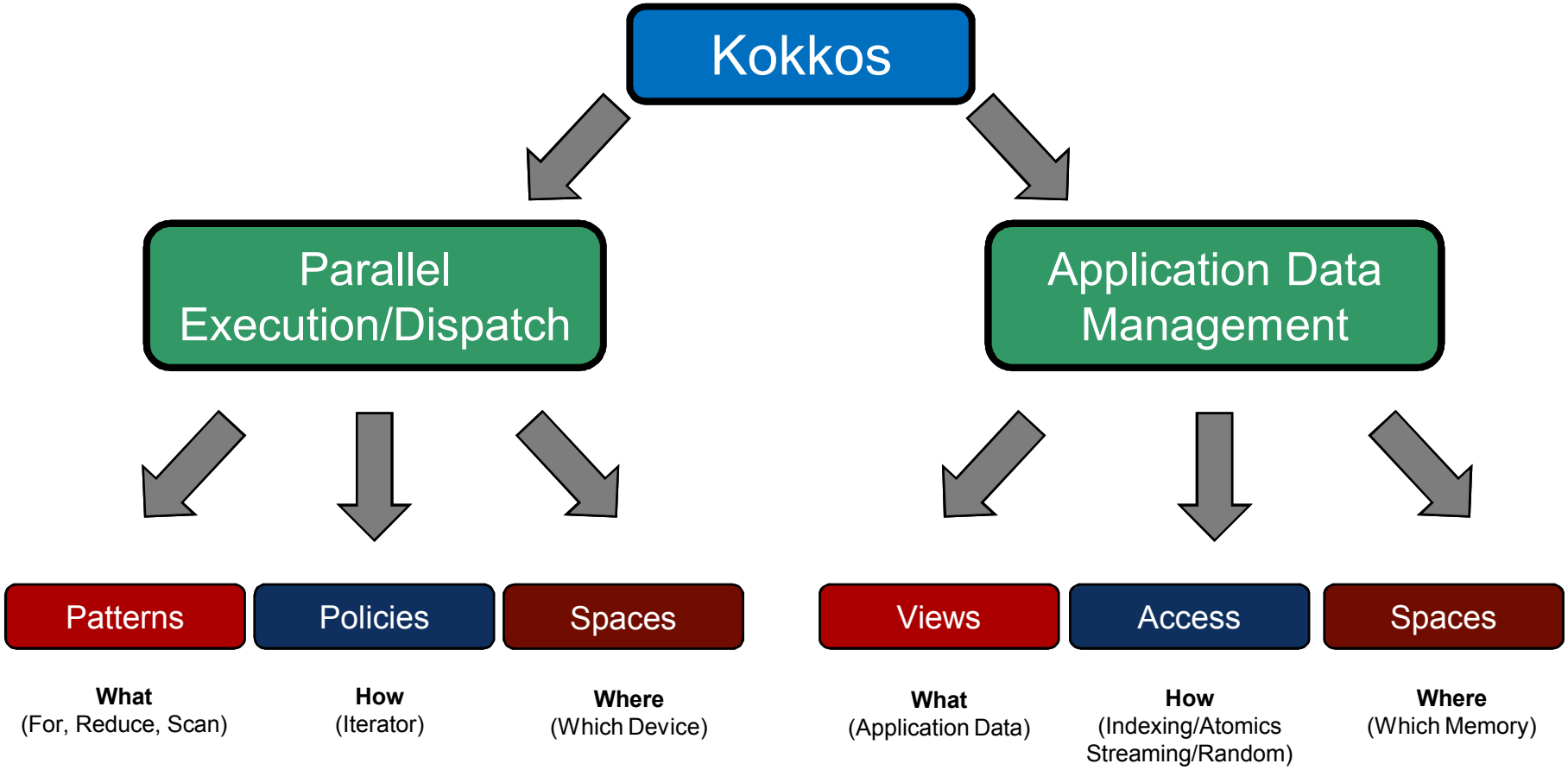
Anything Better?

Sandia National Laboratories

- Sandia has started to develop the **Kokkos Programming Model**
- Why?
  - Abstraction seems to be a good idea today
  - Still not sure what the future will bring?
- Where:
  - http://www.github.com/kokkos

Kokkos Programming Model

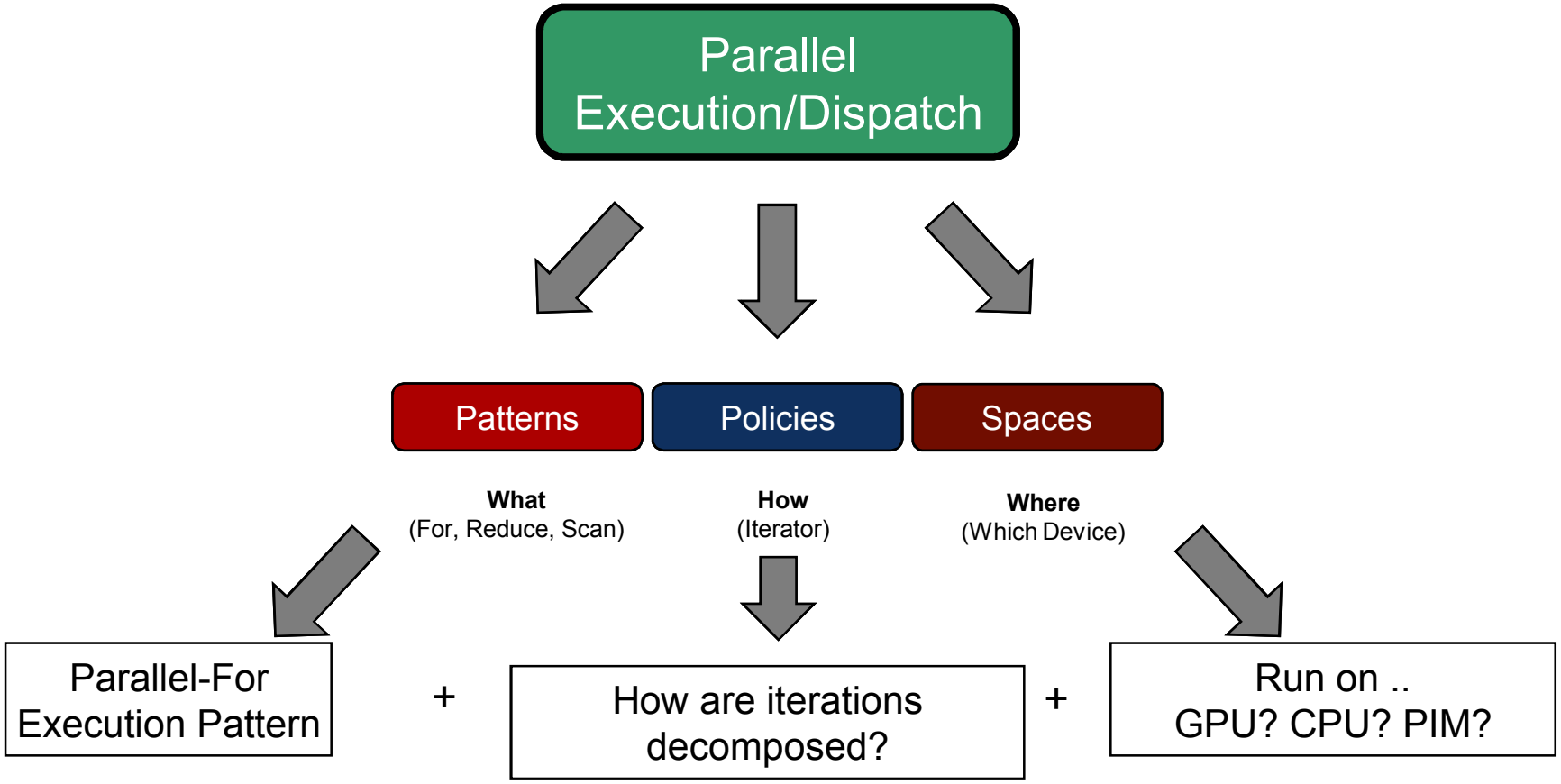
Sandia National Laboratories



Separation and Abstraction of Concerns

Abstract Application Data and Computation

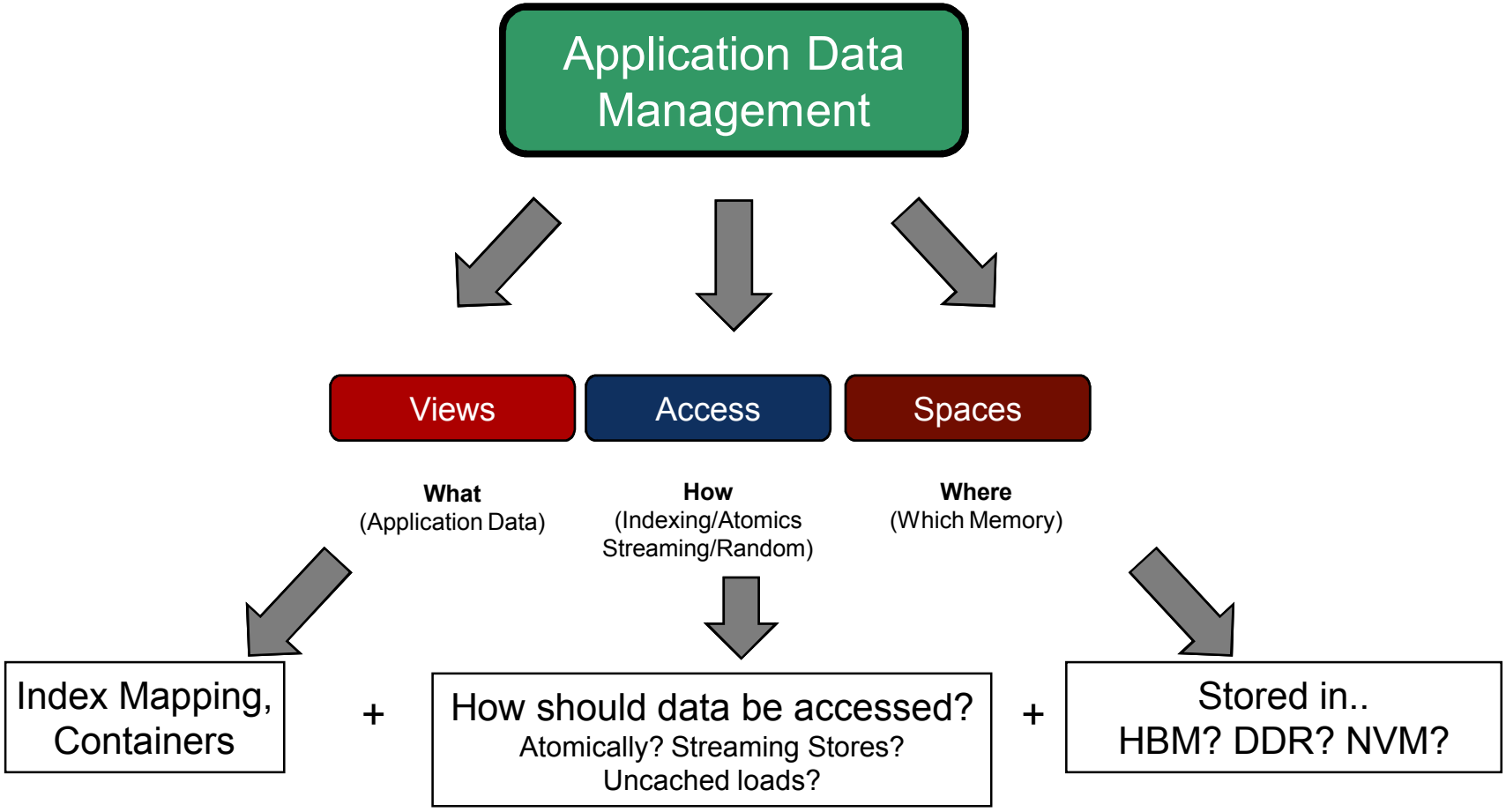
Kokkos Programming Model (Compute)



Sensible defaults for many execution spaces to reduce programmer overhead

Let Sandia research and Kokkos developers handle the heavy work

Kokkos Programming Model (Data)



Sensible defaults for many memory spaces to reduce programmer overhead

Let Sandia research and Kokkos developers handle the heavy work

What Does Kokkos Run on Today?

Sandia National Laboratories

Kokkos is running on every advanced architecture test bed, prototype option on AMD systems

ASC Trinity Phase I – ATS1

- ✔ Intel Xeon Haswell (Intel, GNU, LLVM)

ASC Trinity Phase II – ATS1

- ✔ Intel Xeon Phi Knights Landing Emulator

ASC Sierra – ATS2

- ✔ POWER8 (XL, GNU, PGI)
- ✔ NVIDIA GPU (K20, K40, K80, NSDK-7.5/8.0)

ASC TLCC-2

- ✔ Intel Xeon Sandy Bridge (Intel, GNU, LLVM, Cray)
- ✔ Intel Xeon Broadwell (Intel, GNU, LLVM, Cray)

ASC Advanced Arch. Test Beds

- ✔ AMD Kaveri APU (GNU-HSA)
- ✔ ARM64 (GNU, LLVM)
- ✔ Intel Xeon Phi Knights Corner (Intel)

✔ = Kokkos Build Type in Release ✔ = Prototype/Research

Sandia National Laboratories

CASE STUDY – PORTING LULESH

Examining Porting Strategies for Code Teams

Sandia National Laboratories

- **Very large proportion of ASC code at Sandia is MPI only**
  - Implies a serial on-node model with limited thread safety applied

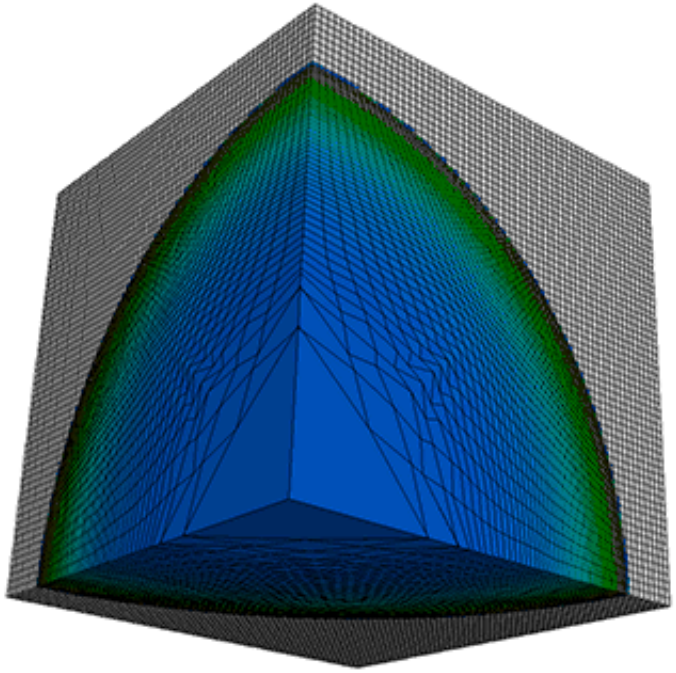
- Starting point for this study is the **serial** version of LULESH
  - Taken from the OpenMP version but with all OpenMP pragmas, reductions and specializations removed ("proxy" for "real" code)

- Provide several implementations to evaluate metrics:
  - **Kokkos:** Minimal CPU, Minimal CPU with ref lambdas, Minimal GPU, Optimized-V1, Optimized-V2, Optimized-V3
  - **OpenMP:** Original OpenMP from LLNL, Optimized OpenMP from SNL
  - **RAJA:** RAJA-Basic and RAJA-Index-Set

LULESH Benchmark

Sandia National Laboratories

- Unstructured hydrodynamics benchmark from Lawrence Livermore
- Somewhat simplified but has nice programming structures relevant to larger codes
- Well studied by vendors and academia
- C/C++
- Implementations in many programming models



https://codesign.llnl.gov/lulesh.php

Non Kokkos-Variants

Sandia National Laboratories

- **RAJA-Basic:** code provided by Jeff Keasler and Rich Hornung from LLNL, uses RAJA abstractions for parallel dispatch
- **RAJA-IndexSet:** code provided by Jeff Keasler and Rich Hornung from LLNL, uses RAJA abstractions for data iteration
- **OpenMP Original:** NO-RAJA variant from LLNL
- **OpenMP Minimal:** a stripped down version using basic parallel-for schemes and atomic operations developed from serial using Intel AdvisorXE and InspectorXE (akin to developer using tools)
- **OpenMP Optimized:** Sandia optimized version which improves vectorization and reduction performance

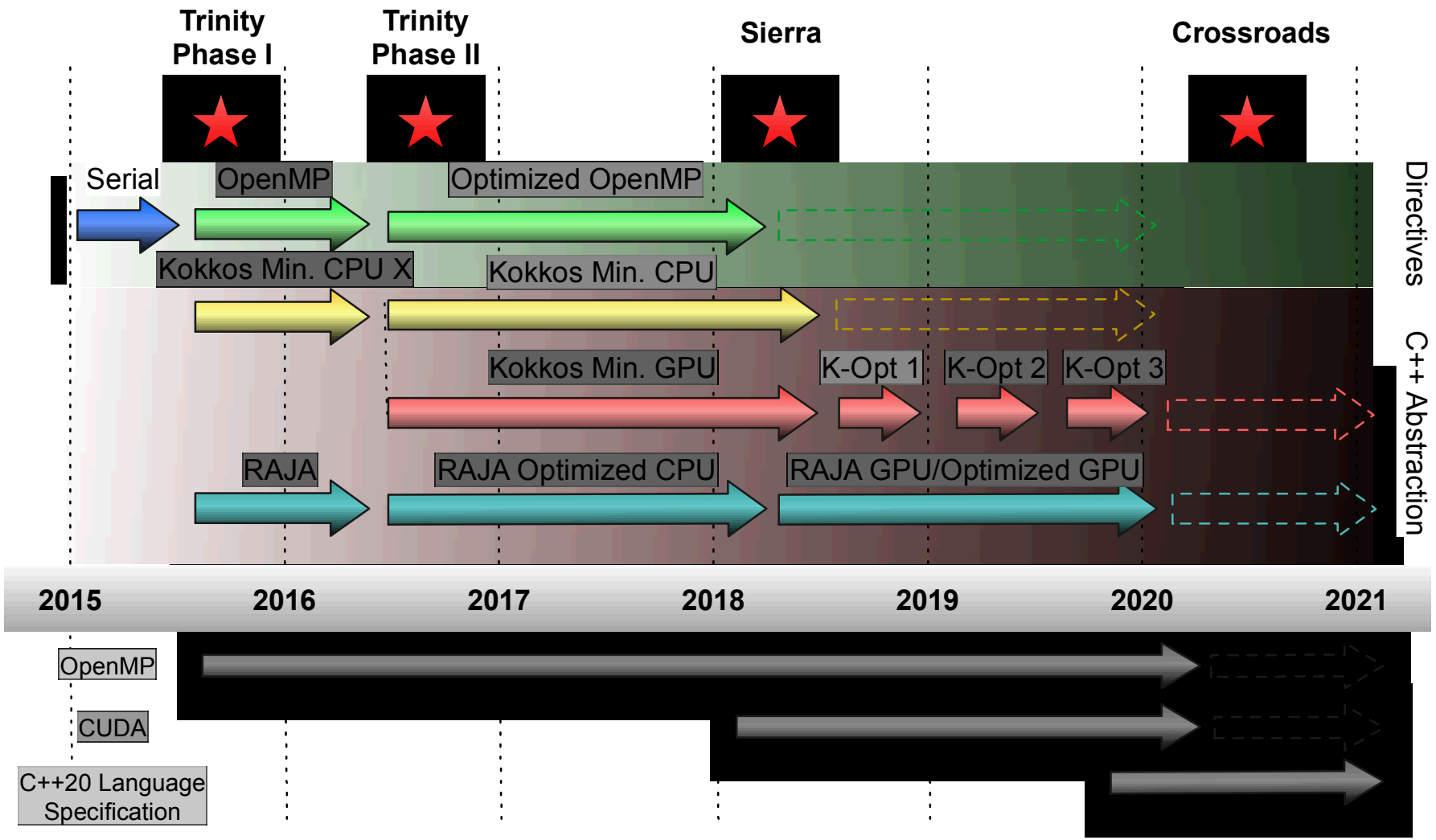
Optimized Kokkos Variants

Sandia National Laboratories

- **Kokkos-Minimal-CPU:** developed by a physicist with limited experience writing threaded code (our experiment for code we would get from many code groups)
- **Kokkos-Minimal-CPU-RL:** basic port to Kokkos which utilizes capture-by-reference lambdas to significantly decrease programmer burden
- **Kokkos-Minimal-GPU:** extension of Kokkos-Minimal-CPU to work on the GPU (mainly data structure const changes)
- **Kokkos-Optimized-v1:** eliminate buffer realloc; reduce register pressure
- **Kokkos-Optimized-v2:** use Kokkos Views with Layout and Traits, Hierarchical Parallelism
- **Kokkos-Optimized-v3:** kernel fusion

Swim Lanes for Code Teams

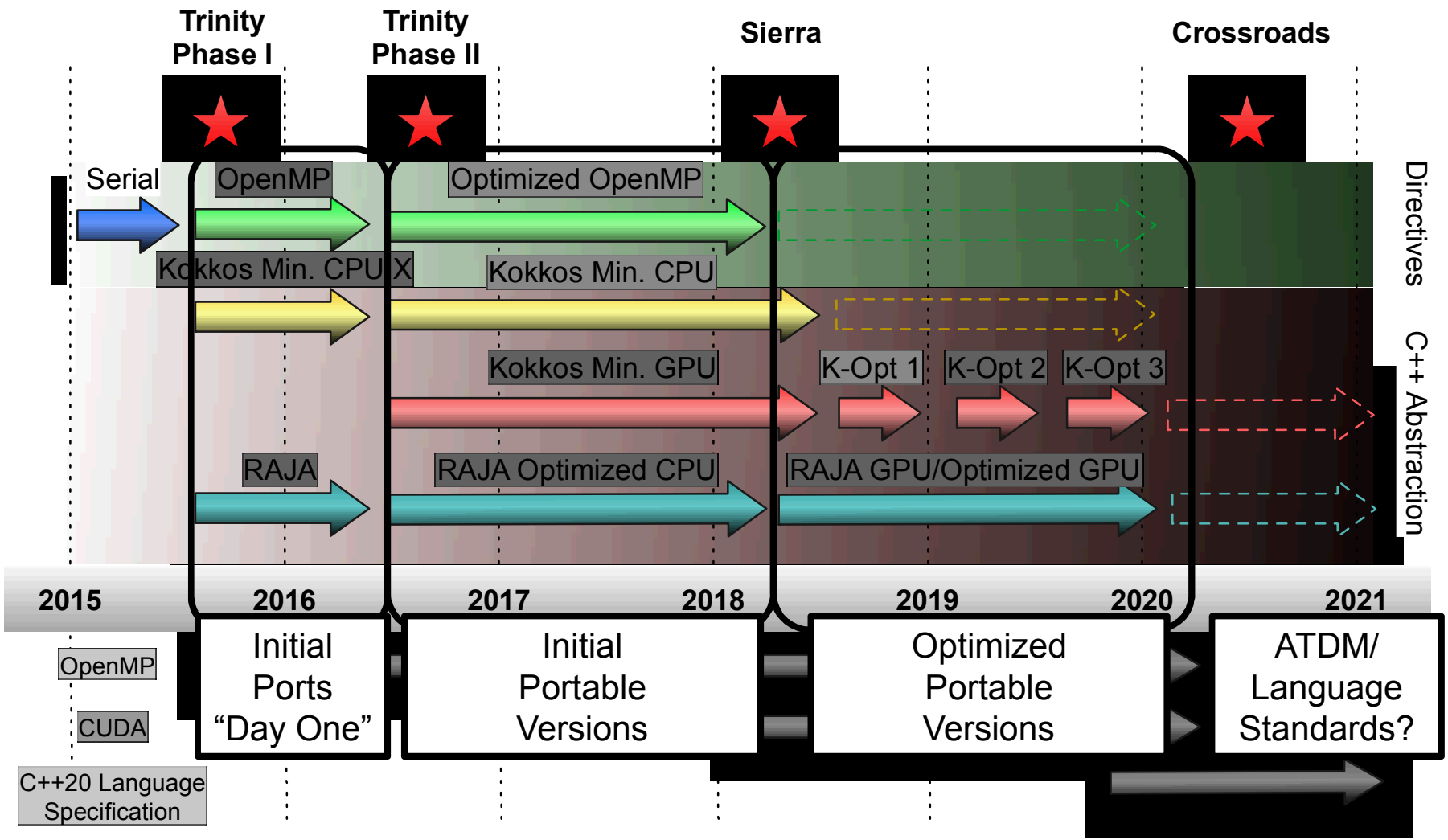
Sandia National Laboratories



This is not an official Sandia position

Swim Lanes for Code Teams

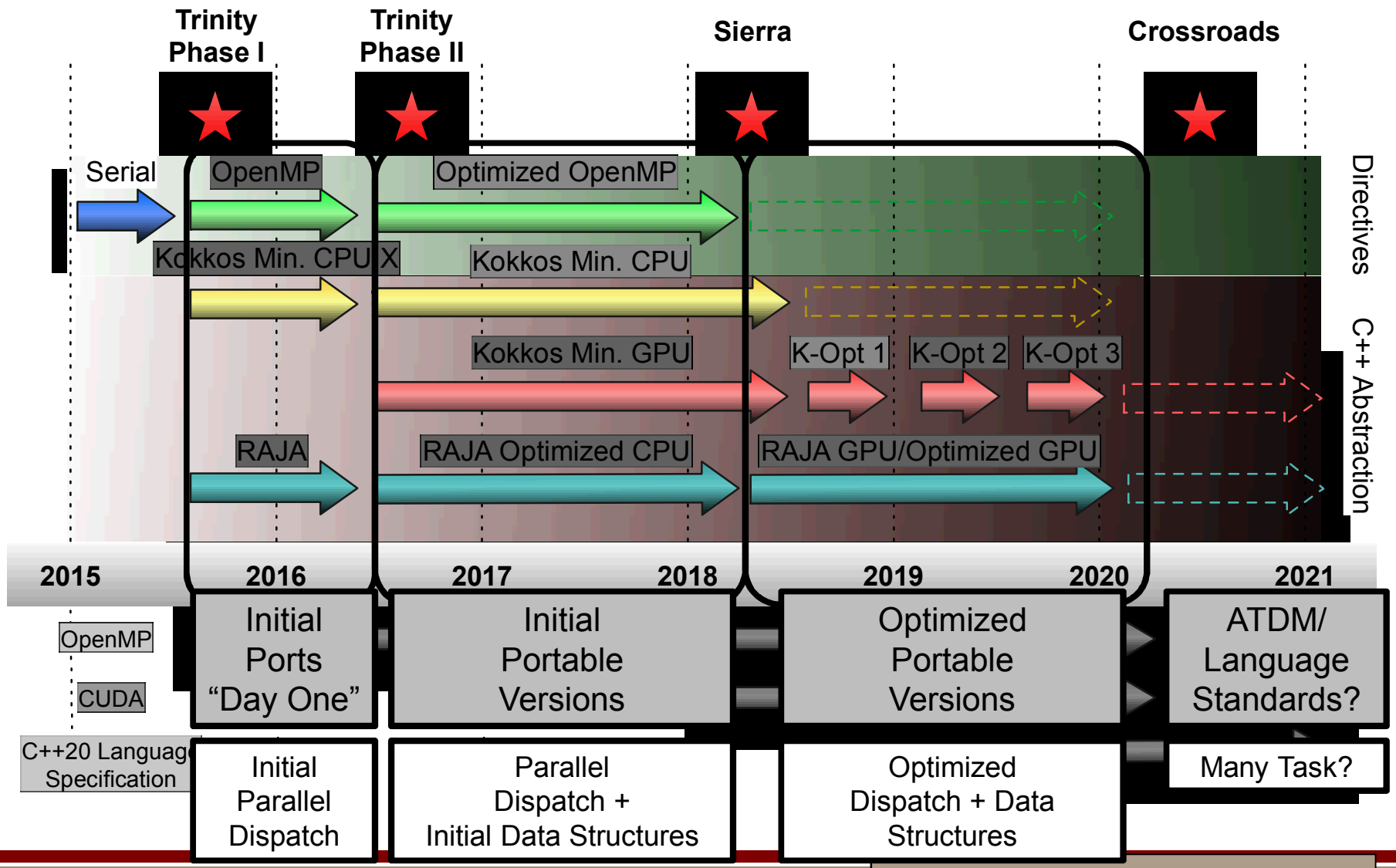
Sandia National Laboratories



This is not an official Sandia position

Swim Lanes for Code Teams

Sandia National Laboratories



This is not an official Sandia position

What are We Presenting?

Sandia National Laboratories

- In an ideal world we would have all code ported with minimal changes
  - Very unlikely to happen for ASC codes, complicated, legacy algorithms, years of engineering

- **So what can we hope for?**
  - Progression of modifications to the code to get them ready for NGP
  - Initial ports require less modification to get code up and running but don't give top performance
  - Slowly evolve code/data-structures to give better cross-platform performance

- Sandia ASC L2 results show what we might be able to expect in a small case study using LULESH
  - We think there is a similar story for Kokkos and RAJA

Sandia National Laboratories

Performance Portability of LULESH Versions

Evaluating Performance Across Architectures

ASC Arch. Test Bed Systems Used For Testing

Sandia National Laboratories

- **Shepard Intel Haswell**
  - Dual-socket, 16-cores/socket, 2 x 256-bit FP-FMA SIMD/core, SMT-2
  - 128GB RAM/socket
  - Intel 15.2.164 Compiler with OpenMPI 1.8.X
- **Compton Intel Sandy Bridge and Knights Corner**
  - Dual-socket 8-cores/socket, 2x256-bit FP SIMD/core, SMT-2
  - 32GB RAM/socket
  - Intel 15.2.164 Compiler with OpenMPI 1.8.X (Sandy Bridge)
  - 57-core KNC-C0, 1.1GHz, 6GB/RAM
  - Intel 15.2.164 Compiler with Intel MPI 4.1.036 (KNC)

ASC Arch. Test Bed Systems Used For Testing

Sandia National Laboratories

- **White POWER8**
  - Dual-socket, Dual-NUMA/socket POWER8, 3.4GHz
  - 5-cores/NUMA = 10 cores/socket = 20 cores/node, SMT-8/core
  - 128GB RAM/NUMA = 512GB/node
  - GNU 4.9.2 with OpenMPI 1.8.X
  - IBM XL 13.1.2 with OpenMPI 1.8.X
- **Hammer APM ARM-64/v8**
  - Single socket/node, 8-cores/node, 2.4GHz
  - 32GB RAM/socket
  - GNU 4.9.2 with OpenMPI 1.8.X

ASC Arch. Test Bed Systems Used For Testing

Sandia National Laboratories

- **Shannon Intel Sandy Bridge + NVIDIA Kepler K40/80**
  - Dual-socket, 8-cores/socket Sandy Bridge = 16 cores/node
  - 32GB RAM/socket
  - NVIDIA Kepler K40 per socket
  - NVIDIA CUDA 7.5 SDK
  - GNU 4.7.2 with OpenMPI 1.8.X (compiled with CUDA support)

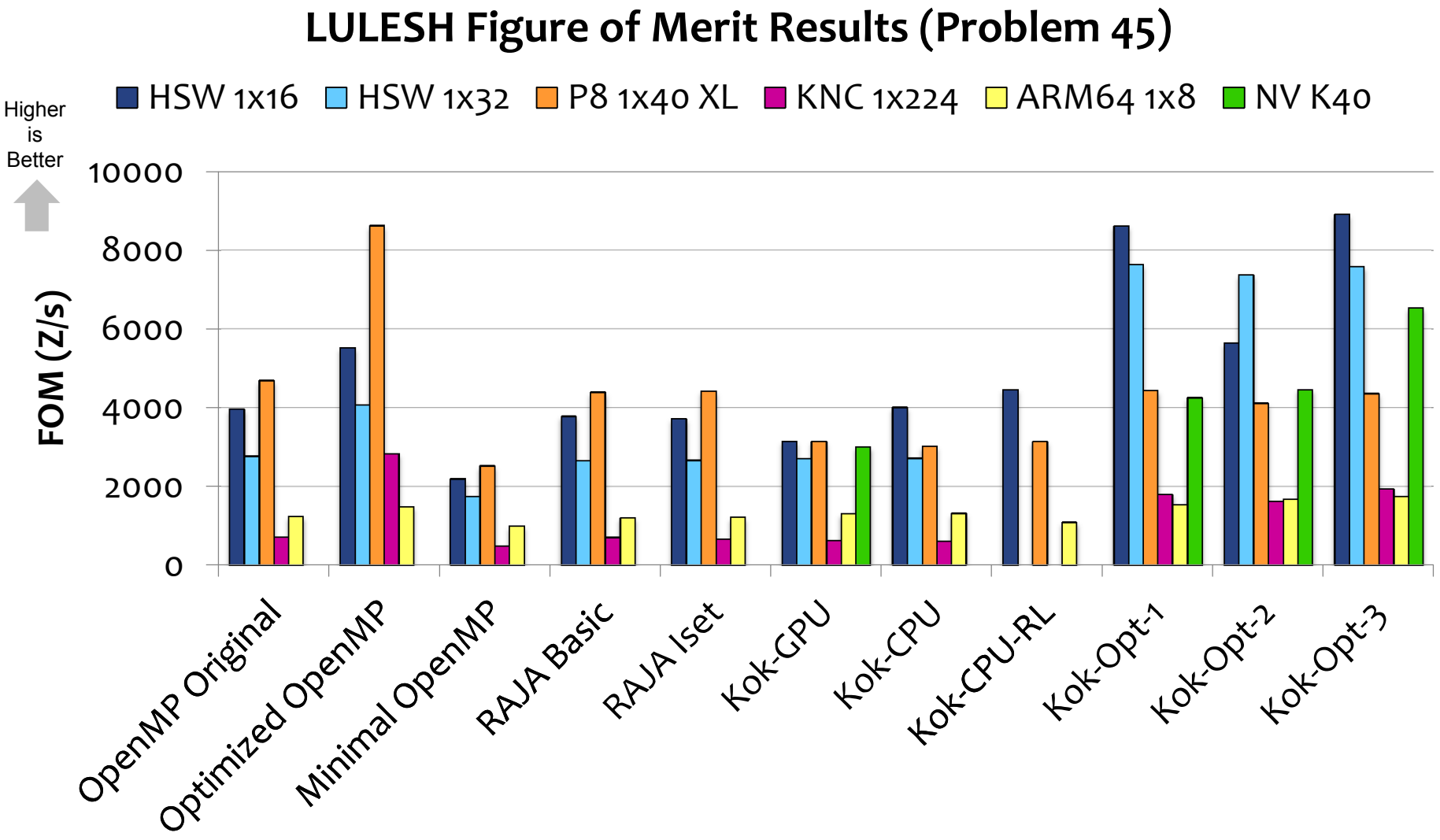
Optimization Notice

Sandia National Laboratories

- Where possible we have selected architecture appropriate optimization flags to improve performance
  - **Kokkos** – baked into the Kokkos Makefile system
  - **RAJA** – baked into RAJA Makefile system and RAJA header files for alignment, vectorization width *etc* (header additions are annoying)
- Results are the harmonic mean of LLNL-coded "Figure of Merit" (FOM) from a minimum 10 runs, max, min etc are all recorded
  - Error bars are typically very small (1-3%) so are not included in plots for brevity
- All configurations used optimized (per platform) MPI process pinning, thread affinities and job configurations
  - Lots of research at Sandia using Mantevo over last four years to understand these issues
  - An on-going process but can give >2X performance difference

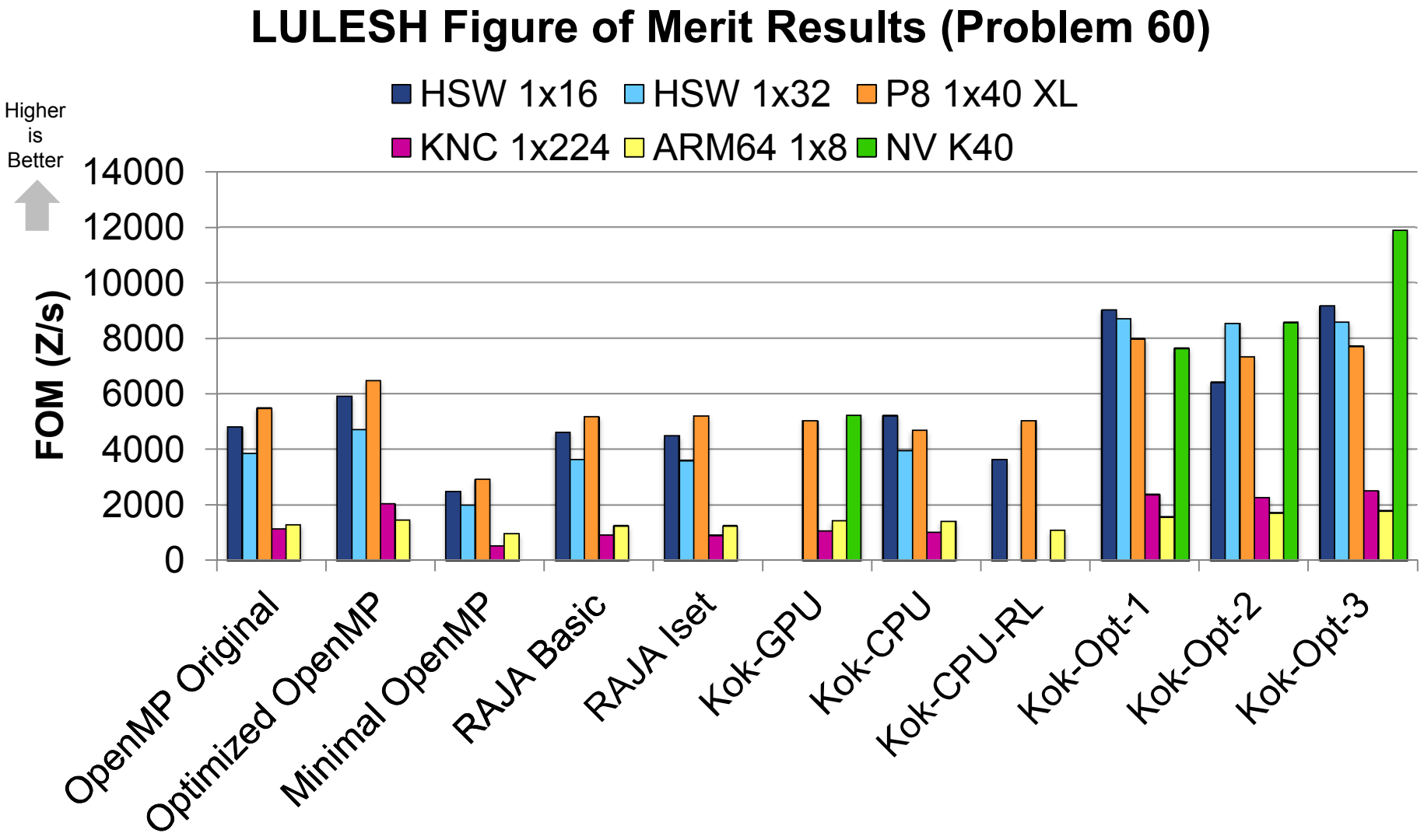
Performance Portability Metrics

Sandia National Laboratories



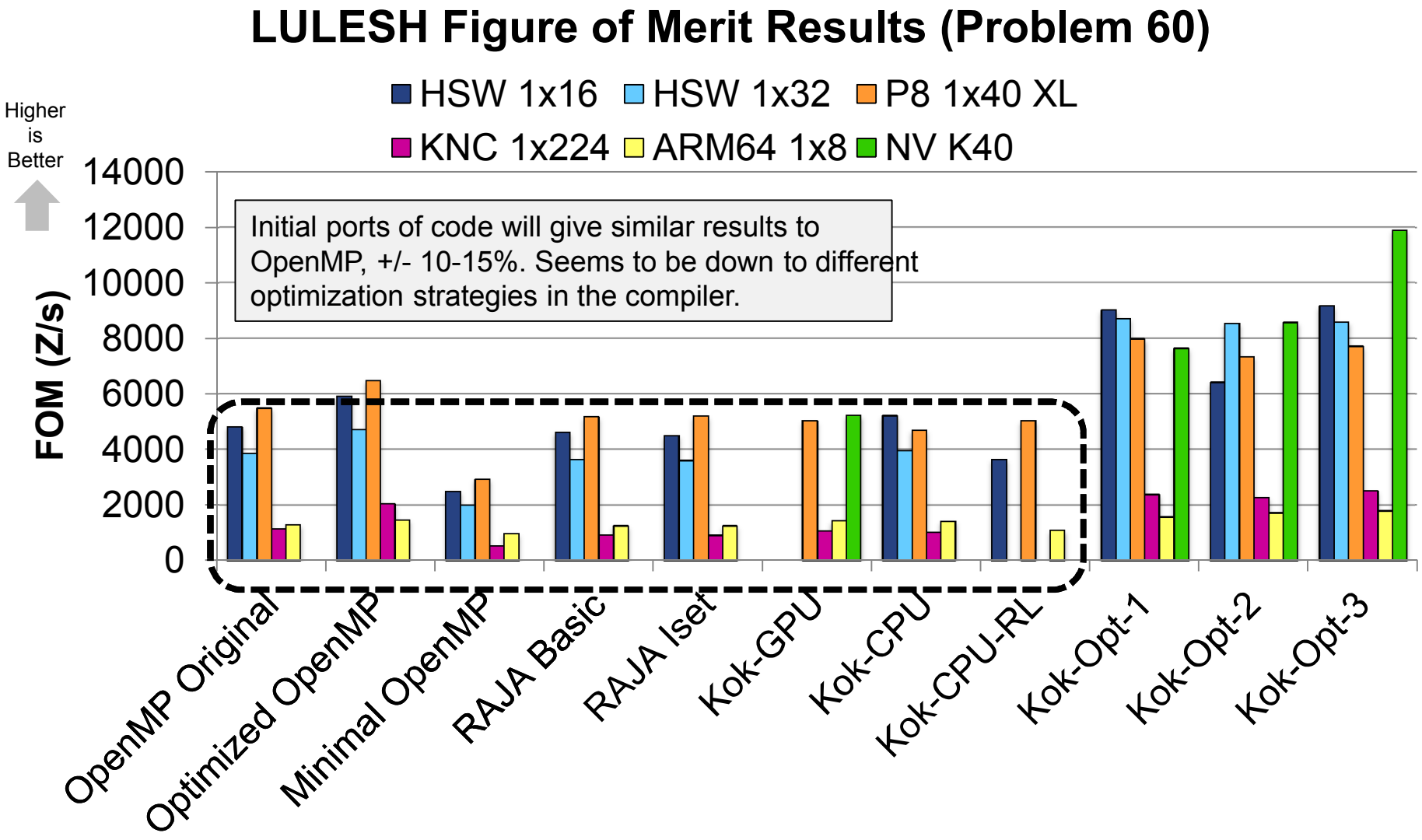
Results by Dennis Dinge, Christian Trott and Si Hammond

Performance Portability Metrics

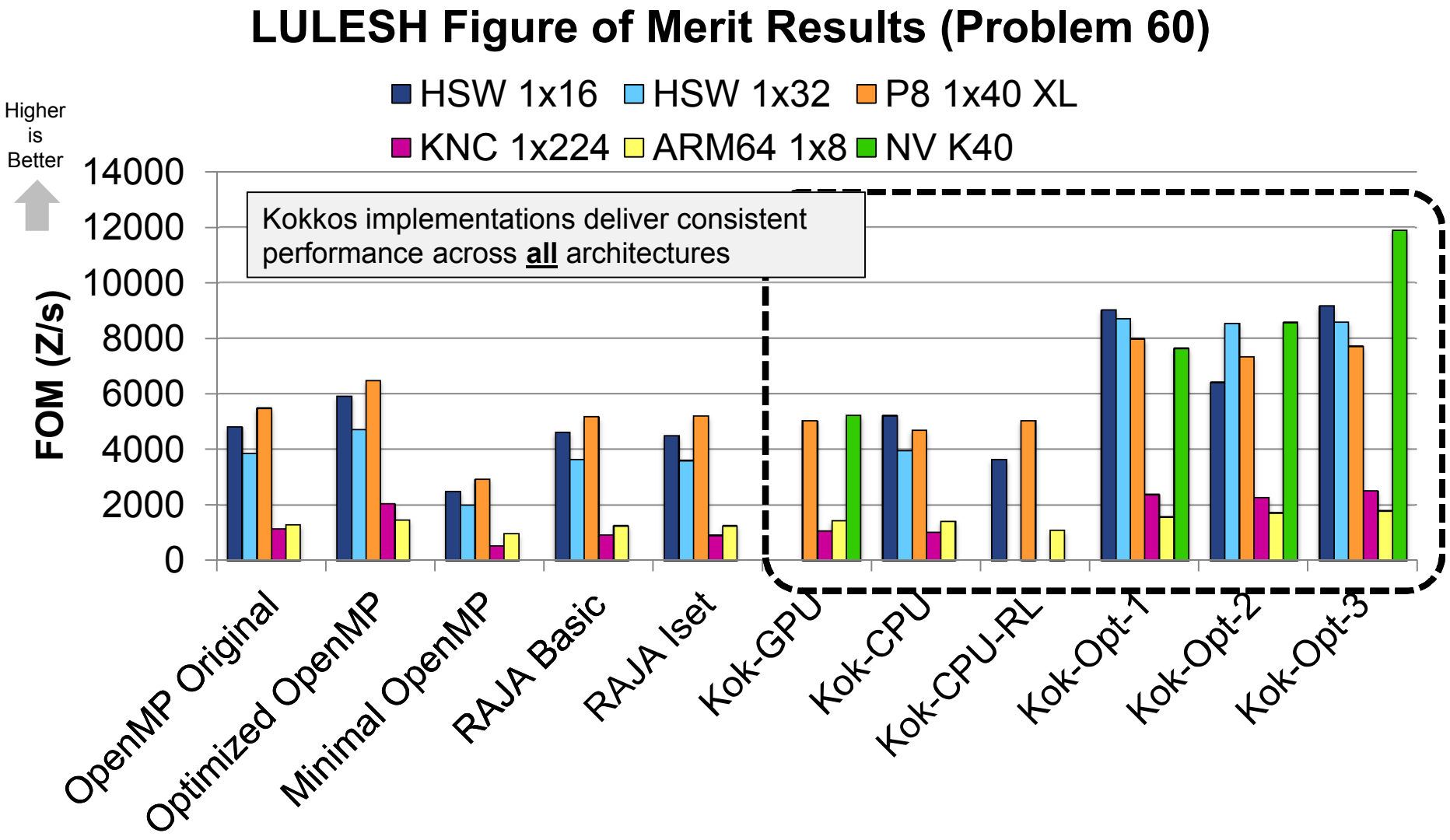


Results by Dennis Dinge, Christian Trott and Si Hammond

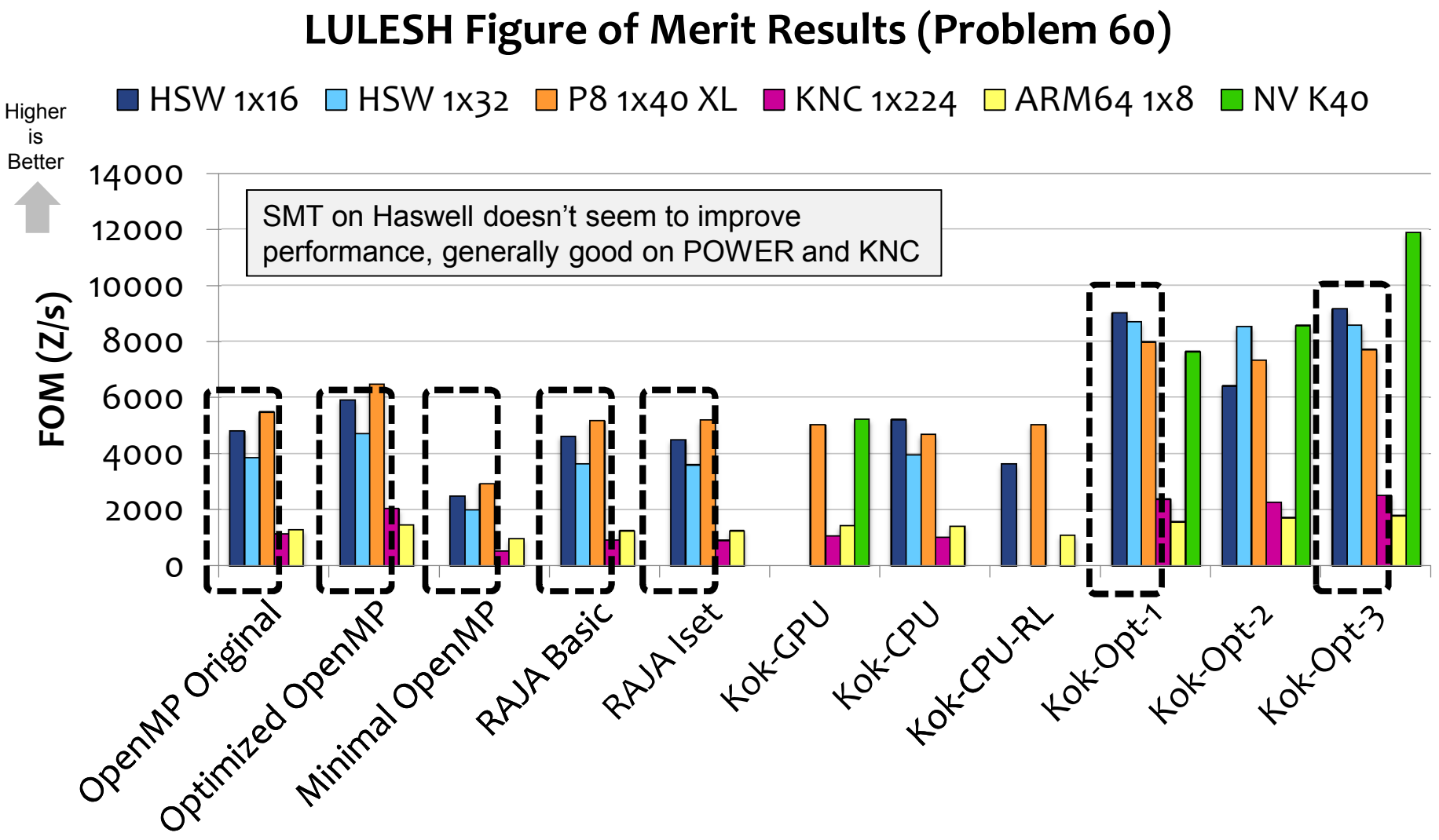
Performance Portability Metrics



Performance Portability Metrics



Performance Portability Metrics



Results by Dennis Dinge, Christian Trott and Si Hammond

Thoughts and Experiences

Sandia National Laboratories

- These problem sizes are small relative to some of the systems
  - O(100) – O(200) MB in problem size
  - POWER8 – very large memory, large caches (particularly L4)
  - GPU – needs more parallelism

- We are trying to capture performance effects based on feedback from LULESH developers
  - But larger problems help our optimizations even more

- Not necessarily demonstrating the best potential FOM performance
  - Can get up to 2X these FOM figures from our implementations

Sandia National Laboratories

Programmer Productivity of LULESH Versions

Evaluating Effort to Develop Versions using Performance Portable C++ Abstraction Layers

How do we calculate “productivity”?

Sandia National Laboratories

- With great difficulty – lots of discussion in the community about what this *really* means

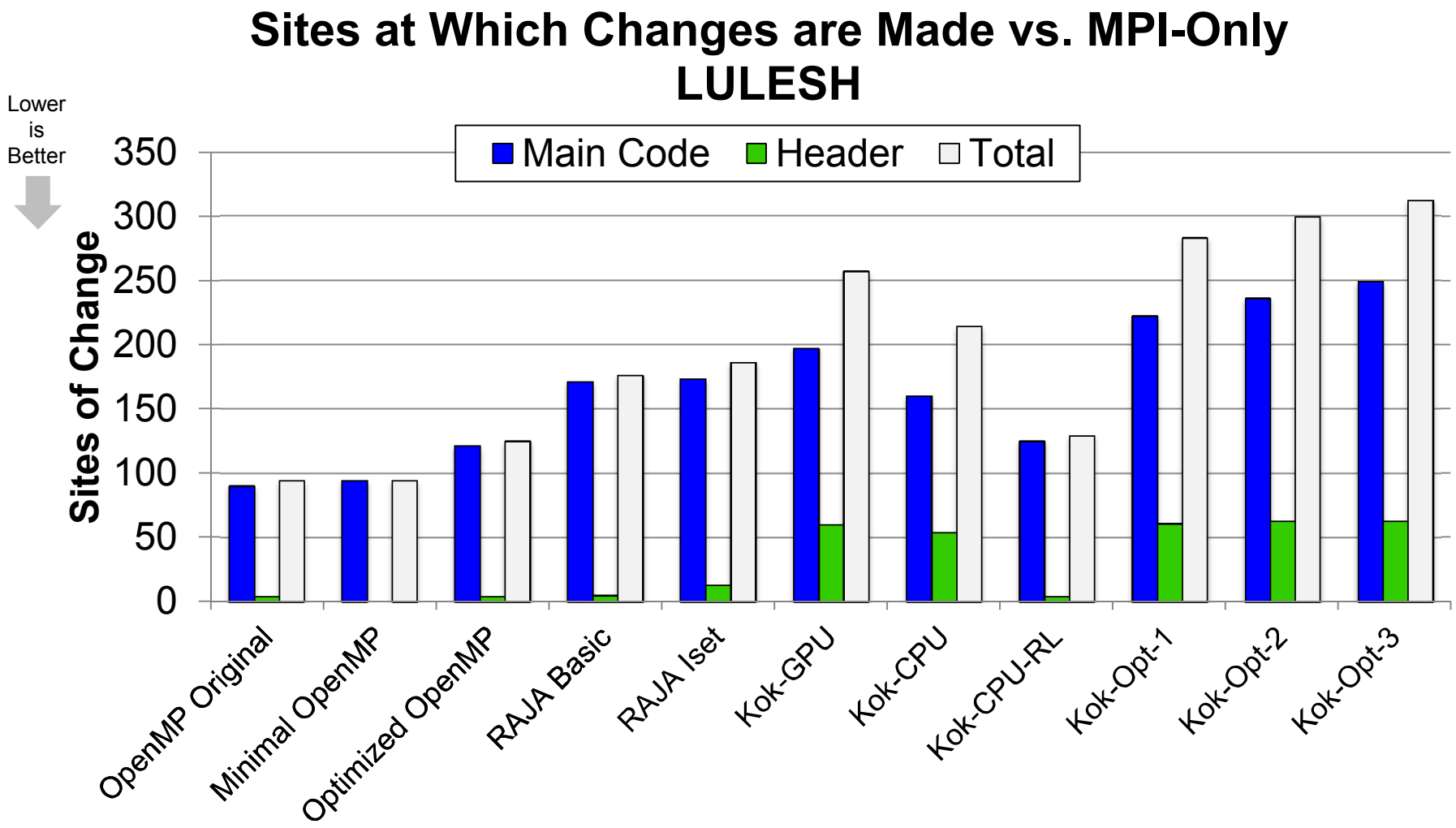
- Our approach:
  1. Remove all comments from the code
  2. Utilize the clang-format LLVM tool with “Google” code option
  3. Compare the number of sites using Apple’s FileMerge tool
  4. Compare the lines added/removed using diff –b –w <paths>

- Not perfect and we have hand modified code of *all* versions to bring the counts more into line (and to be fair wherever possible)
- Point is to show approximate level of programmer effort not be precisely quantitative because coding style largely down to individual

http://clang.llvm.org/docs/ClangFormat.html

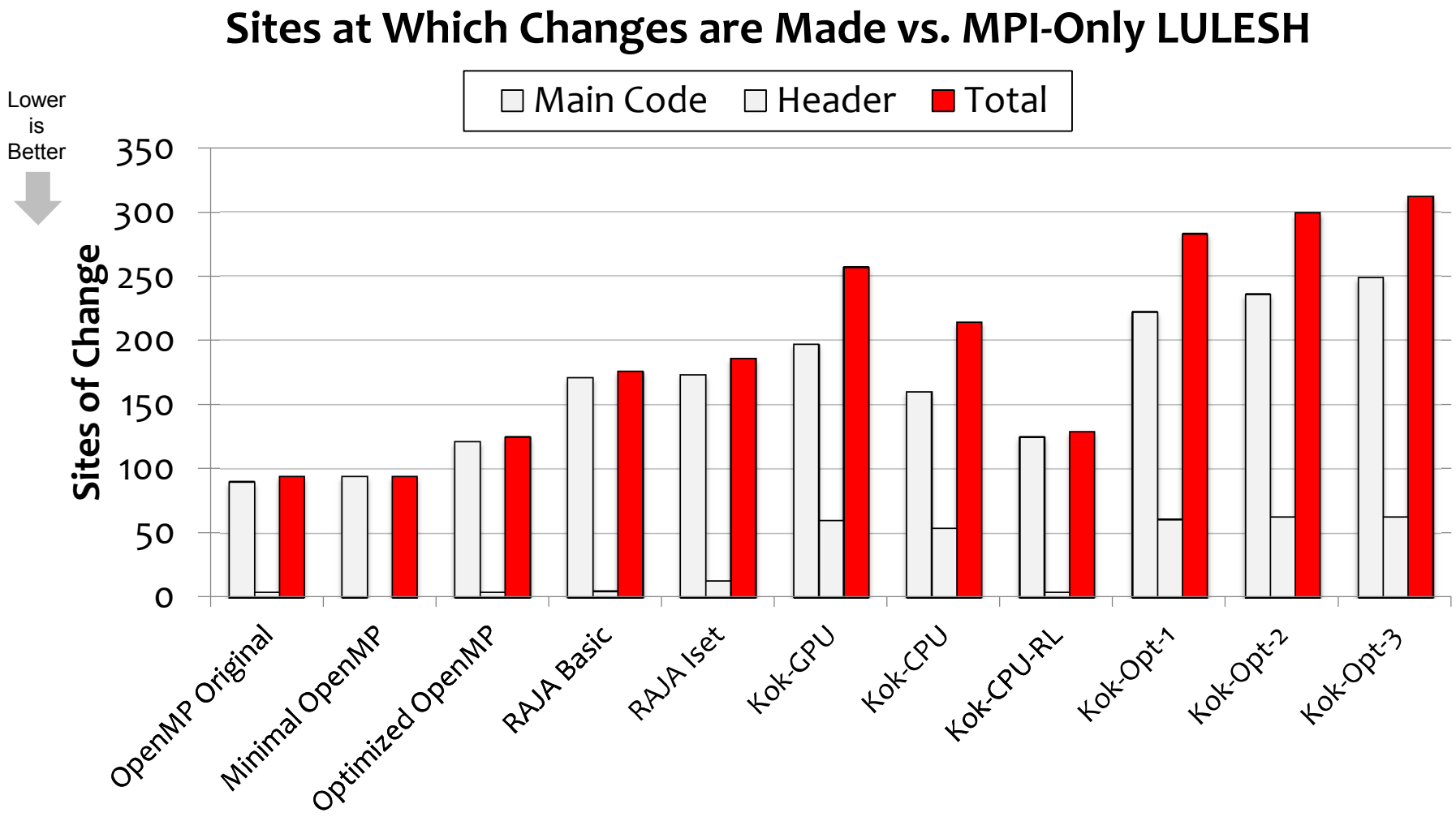
Count of Sites at Which Changes are Made

Sandia National Laboratories



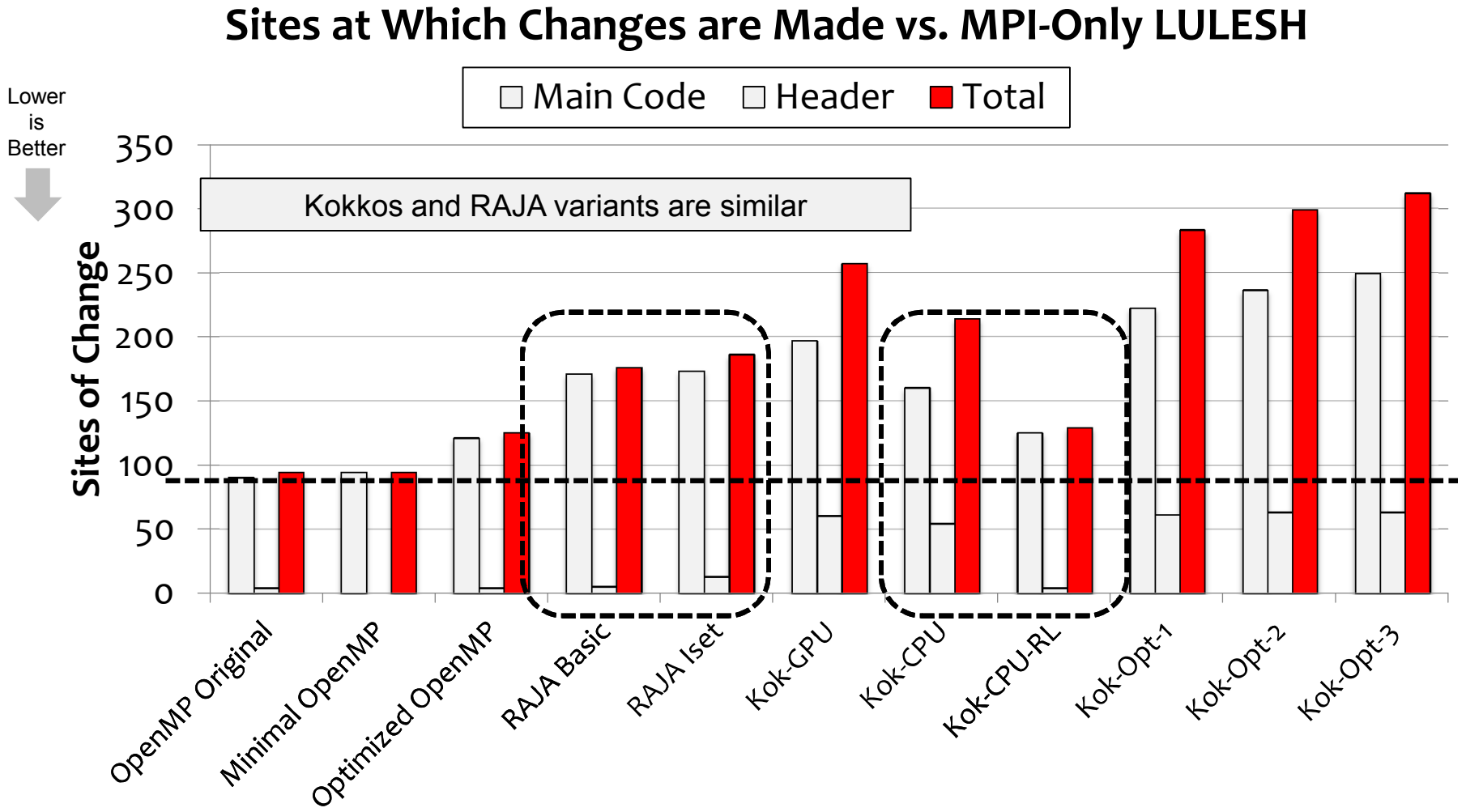
Count of Sites at Which Changes are Made

Sandia National Laboratories



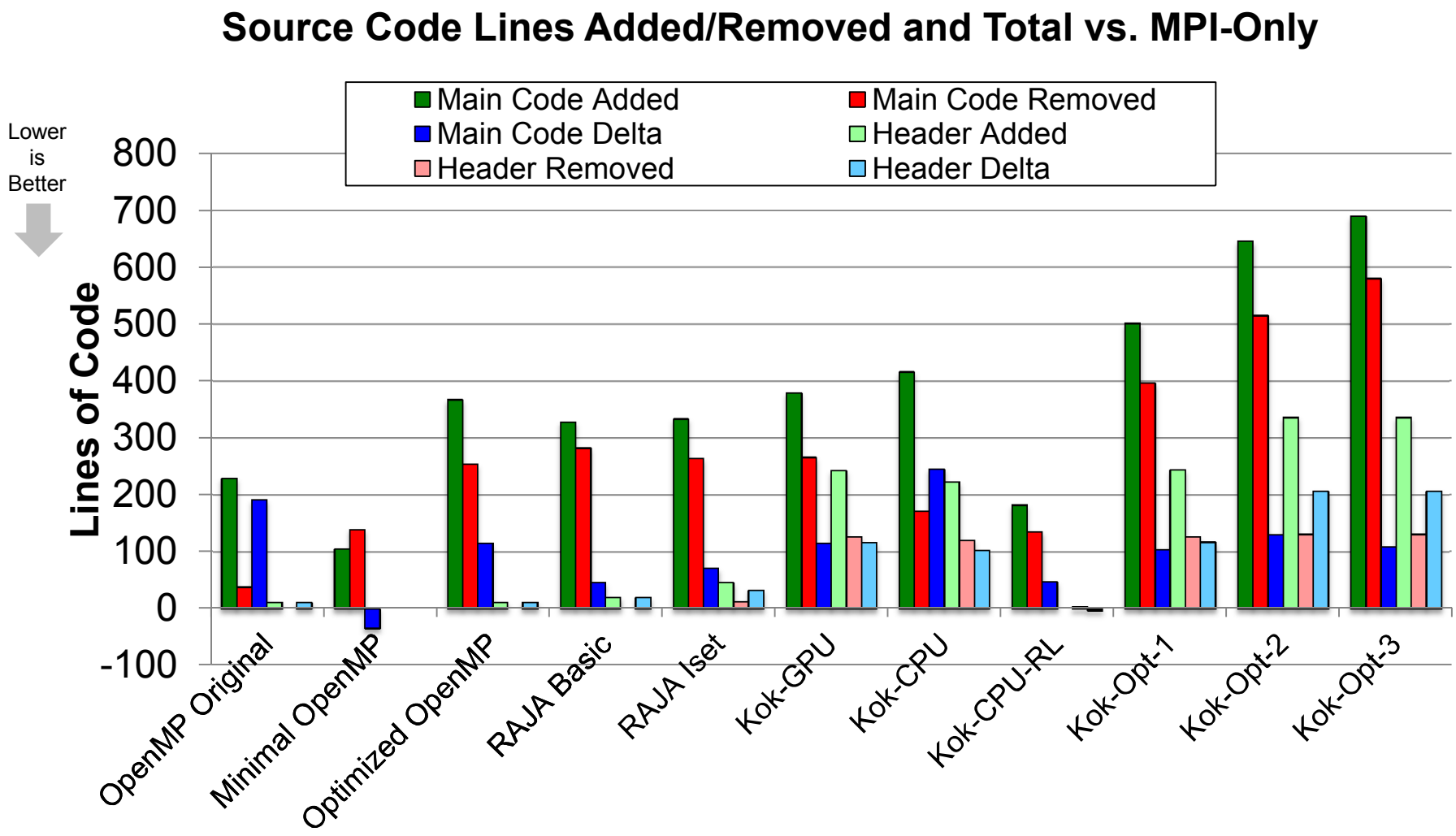
Count of Sites at Which Changes are Made

Sandia National Laboratories



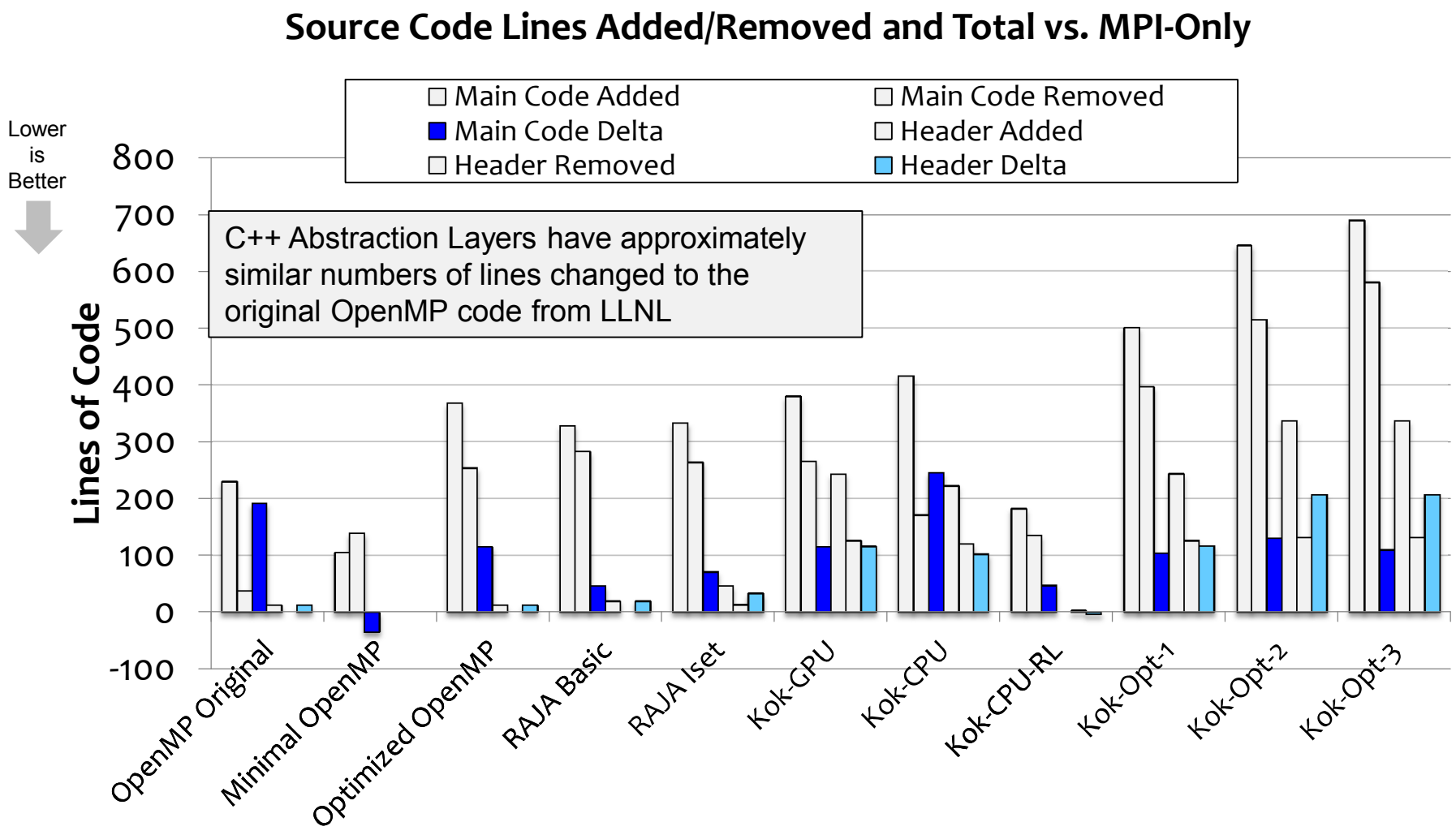
Source Code Line Changes

Sandia National Laboratories



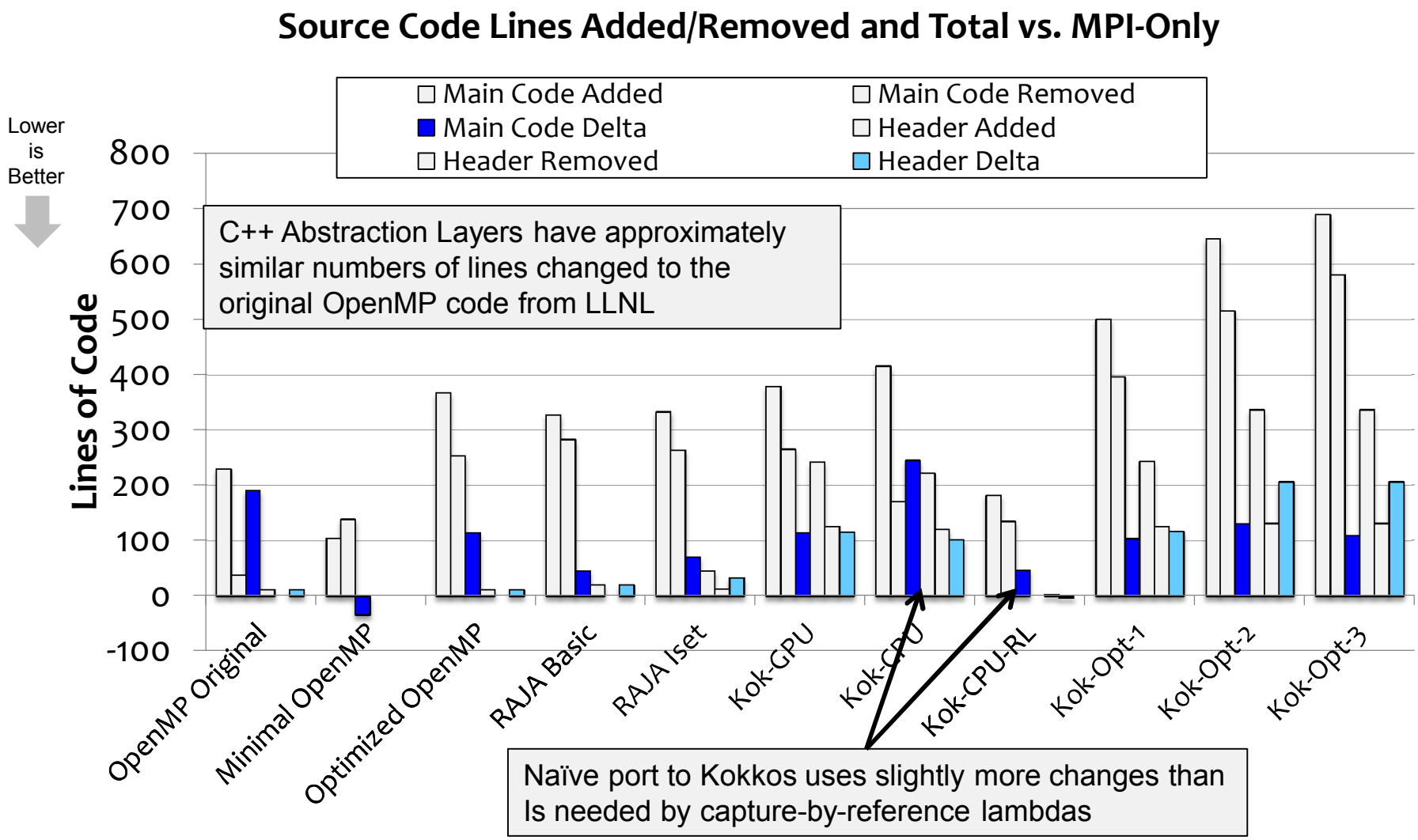
Source Code Line Changes

Sandia National Laboratories



Source Code Line Changes

Sandia National Laboratories



Programmer Development Time

Sandia National Laboratories

- Initial Kokkos-CPU port by a competant physicist took a few months
  - No threading/OpenMP/Kokkos experience for code development
  - Lots of correctness and performance issues came up
  - Initial experience with programmer tools and profilers
- Kokkos optimized implementations
  - O(few weeks) of a Kokkos expert's time
- OpenMP initial and optimized implementations
  - O(few days - week) of a performance analysts time written on a plane
- These are not significant amounts of FTE but the code is small in comparison to production settings (but code groups are larger and better resourced)
- Difficult (impossible?) to do a deep quantitative comparison

What can we take away?

Sandia National Laboratories

- C++ abstraction layers are using similar numbers of changes in code (both code sites and SLOC-delta) to directives

- Perhaps to be expected given implementation strategy is similar in unoptimized variants of the code
  - This is a good thing for developers – hard work is in developing the parallel algorithm, not in how it is expressed in source code

- Looking at changing roughly 15% of the code to get initial parallel versions in this example
  - Warning: example is friendly to parallelism because of its heritage

- Do we need directives in application code at all?

Kernel Analysis for Kokkos Applications

Sandia National Laboratories

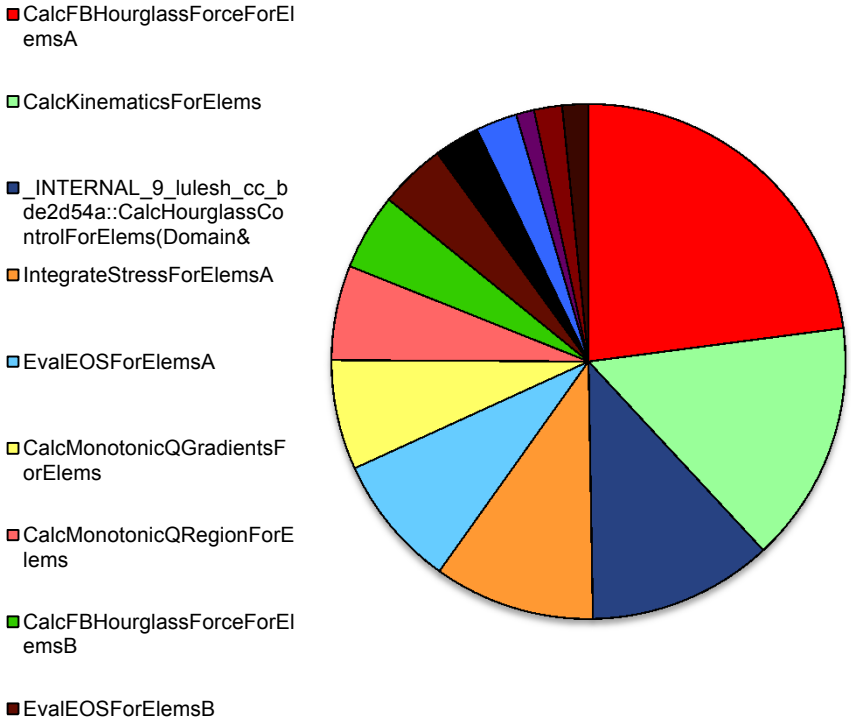
- **Consistent profiling across architectures is hard**
  - Vtune does not like to profile deep in OpenMP hierarchies which are enclosed in headers
  - Nsight manages OK
  - Not clear that tools understand C++ abstraction layers

- **KokkosP Profiling Layer**
  - Recent addition to Kokkos, option to always compile in
  - Tools dynamically loaded, can be stacked, lightweight
  - Expose calling structure of kernels and devices to profiler
  - Better context awareness of what execution is being requested
  - Still very early prototype but shows some promise

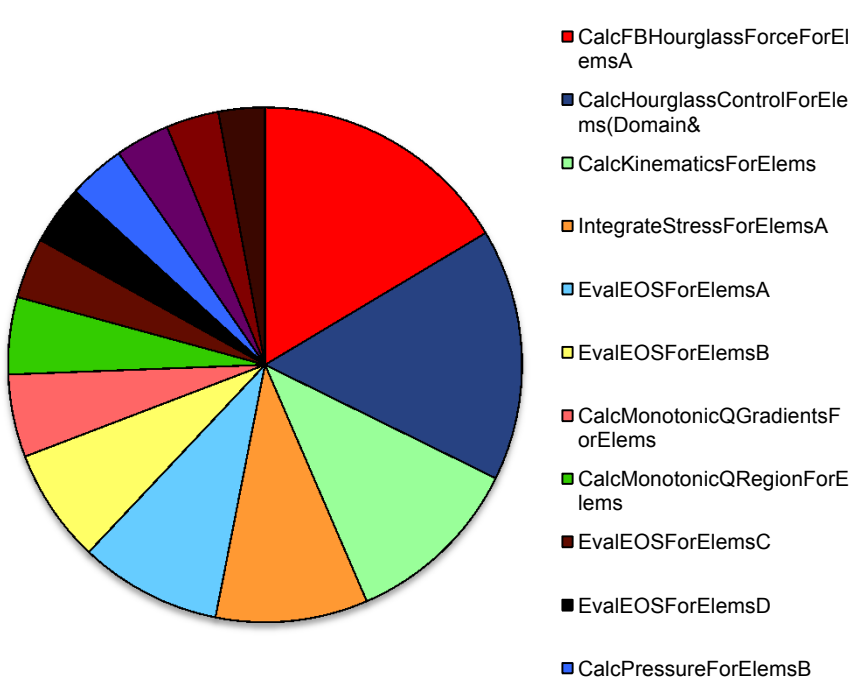
KokkosP Kernel Comparison of Kokkos Opt 1

Sandia National Laboratories

Haswell 1x16 S=45 I=1000



POWER8 1x40 S=45 I=1000



See similar breakdown across architectures but we can profile them all using one tool

Some Thoughts and Discussion

On the Long Road Ahead…

The Long Road Ahead

Sandia National Laboratories

- Many of our codes represent 20+ years of development and continuous refinement
  - We have invested $Bn's into each one because they add real value to our scientific delivery
  - They are complicated and can't be easily replaced
- Tempting to think that new algorithms, new programming models *etc* mean we need to burn down the houses and start over
  - Perhaps a good reason to do it
  - But, doing so will take a *long* time and cost *a lot* of money
- The only really practical path is something which allows a gradual migration of codes and libraries

The Long Road Ahead…

Sandia National Laboratories

- Some see this as a pessimistic view but I see the opportunities
  - Develop abstractions that are “forward-flexible”
  - Don’t commit us to specific hardware *or* software in the future
- Immense performance opportunities as we do go through modifying our kernels
  - Seeing huge performance improvements today in our rewrites
- In the end the hardware and the power/energy are *very* cheap in comparison to programmer’s time and the *value* of the science they produce
  - Is 20M$ = 20MW such a great goal versus >$Bn of programmer time per application?

But here's *an* opinion…

Sandia National Laboratories

- The programming models of the future are probably…
  - The ones which work relatively well with existing languages
  - Are productive
  - Straight forward concepts without all the "bells and whistles"

- The programming models which win developer's hearts are easy to profile, easy to debug, concise

- **Workshops like this can have a huge impact on where we go because they allow us to explore *potential***

- Exciting time to be in the programming models community
  - Perhaps the most exciting time for two decades?

Exceptional service in the national interest

sdhammo@sandia.gov