Unclassified Unlimited Release SAND2015-9079C

Exceptional service in the national interest

Sandia National Laboratories

SoC4HPC – An On-Ramp for Applications at Exascale?

S.D. Hammond
sdhammo@sandia.gov

Scalable Computer Architectures
Center for Computing Research
Sandia National Laboratories, NM, USA

U.S. DEPARTMENT OF ENERGY NNSA National Nuclear Security Administration

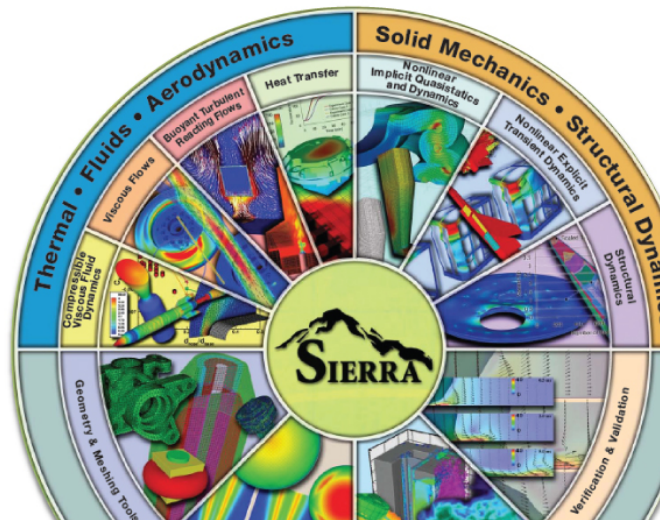
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Mini-Overview of Sandia

- National Laboratory with sites in across the country (DOE, DoD, Industry etc)

- Part of the NNSA Trilab complex associated with ensuring safety of the nations nuclear arsenal (Sandia focused on engineering)

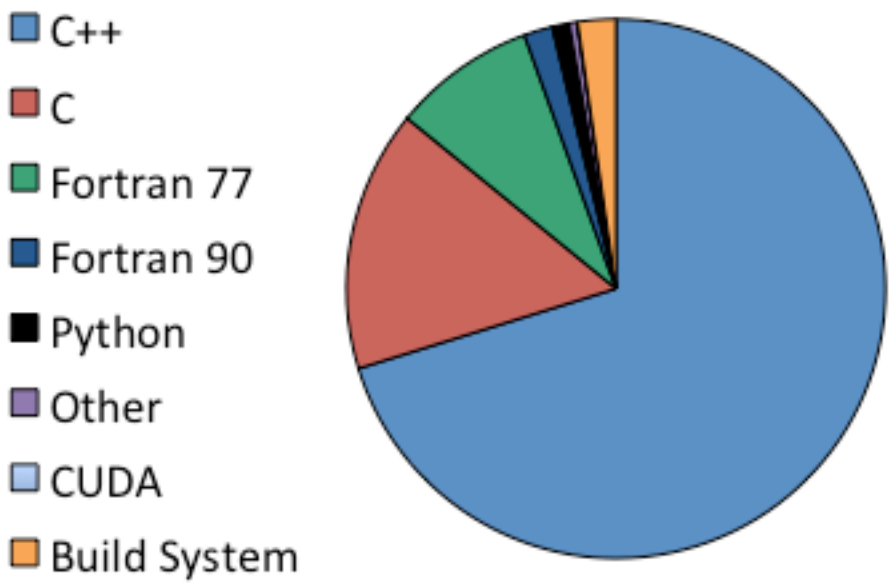
- We do *much* more
  - Leadership in wide range of engineering
  - Supports complex data analytics research
  - Renewable energy
  - Partnerships with industry
  - Systems for space/satellites/hostiles
  - Strong mathematics research
  - Quantum computing and novel devices
- **All supported by broad HPC requirements**



What is the Scale of Our Applications?

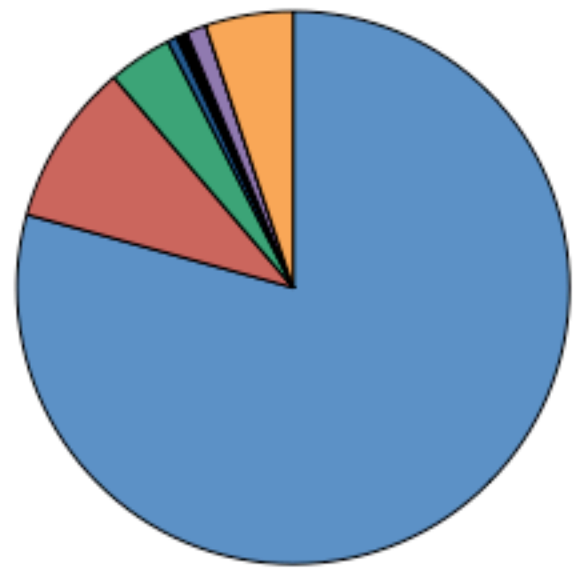
Sandia National Laboratories

Several Sandia Engineering Applications



~11.6M **<u>Application</u>** Lines of Code
(Several Applications, Much Shared)
>50 Third Party Libraries

Sandia Mathematics / Solvers TPL



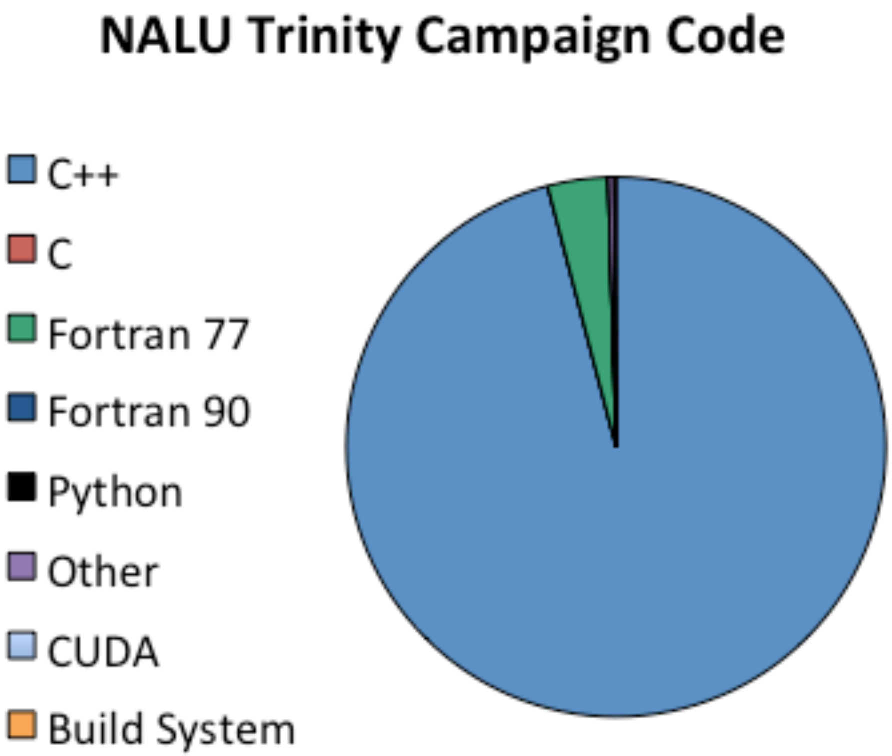
~4.2M Lines of Code
(Very Large Proportion Shared)

This is just a **<u>small</u>** part of our application portfolio

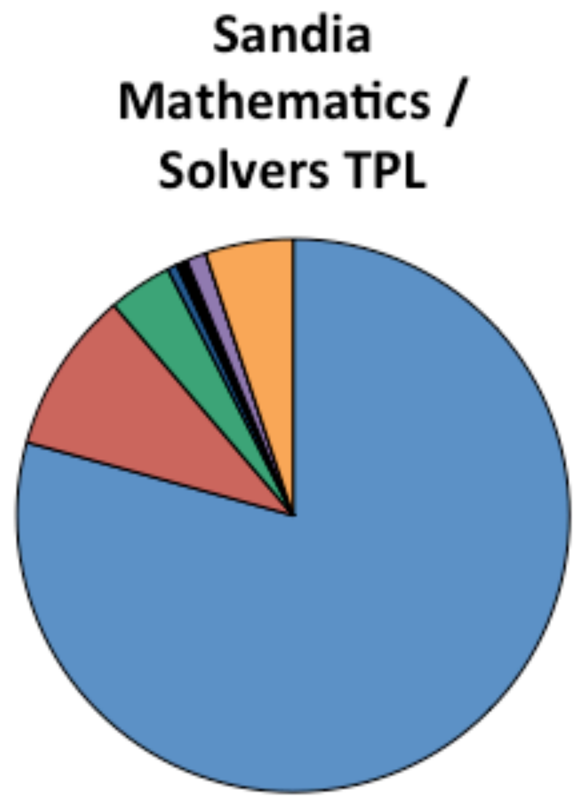
https://github.com/trilinos/trilinos

This is lines of code, does not include comments, white space, documentation etc, no meshing, viz, analysis *etc*

Typical Single Physics Research Codes



~65K **<u>Application</u>** Lines of Code
>5 Third Party Libraries



~4.2M Lines of Code
(Very Large Proportion Shared)

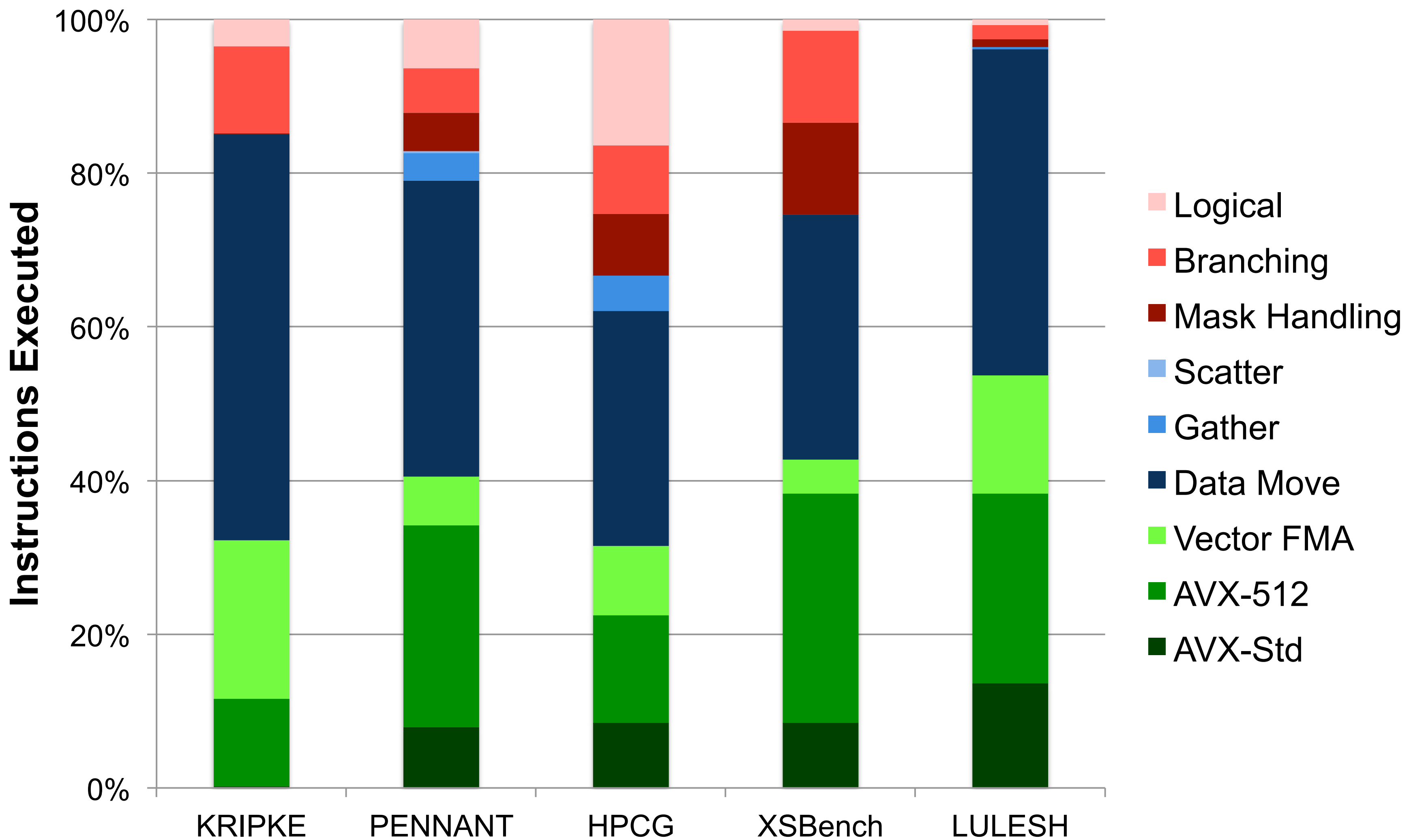
https://github.com/spdomin/Nalu

https://github.com/trilinos/trilinos

Challenges

- **The size and complexity of these codes is a significant challenge (multiple millions of SLOC)**
  - Complexity is very high, written by world class specialists in their field
  - Some of algorithms and techniques are not well documented in literature
  - Some of the code is old, well trusted
  - Analysts demand high reproducibility
- **Varied problem scales and processor cores**
  - Depends on use cases
  - Creates pressure to optimize for weak and strong scaling
- **Challenging to move the code base to new architectures quickly, easily and accurately**
  - *Need* to do so in order to cope with demands from users

Instruction Family Breakdown



Codes compiled for KNL with MiniMPI, Intel 15.1 Compiler, AVX512-MIC Optimization, No Code Optimization Applied, Instructions show for OMP_NUM_THREADS=1

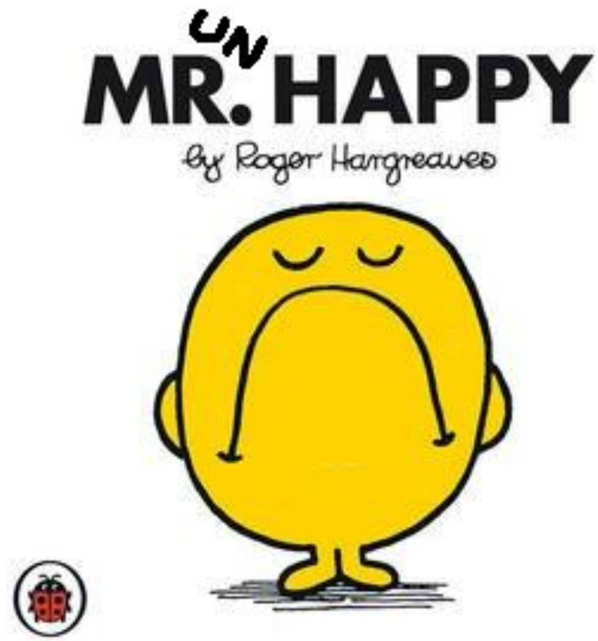
“Wow, Aren’t you Guys Screwed?”

Sandia National Laboratories

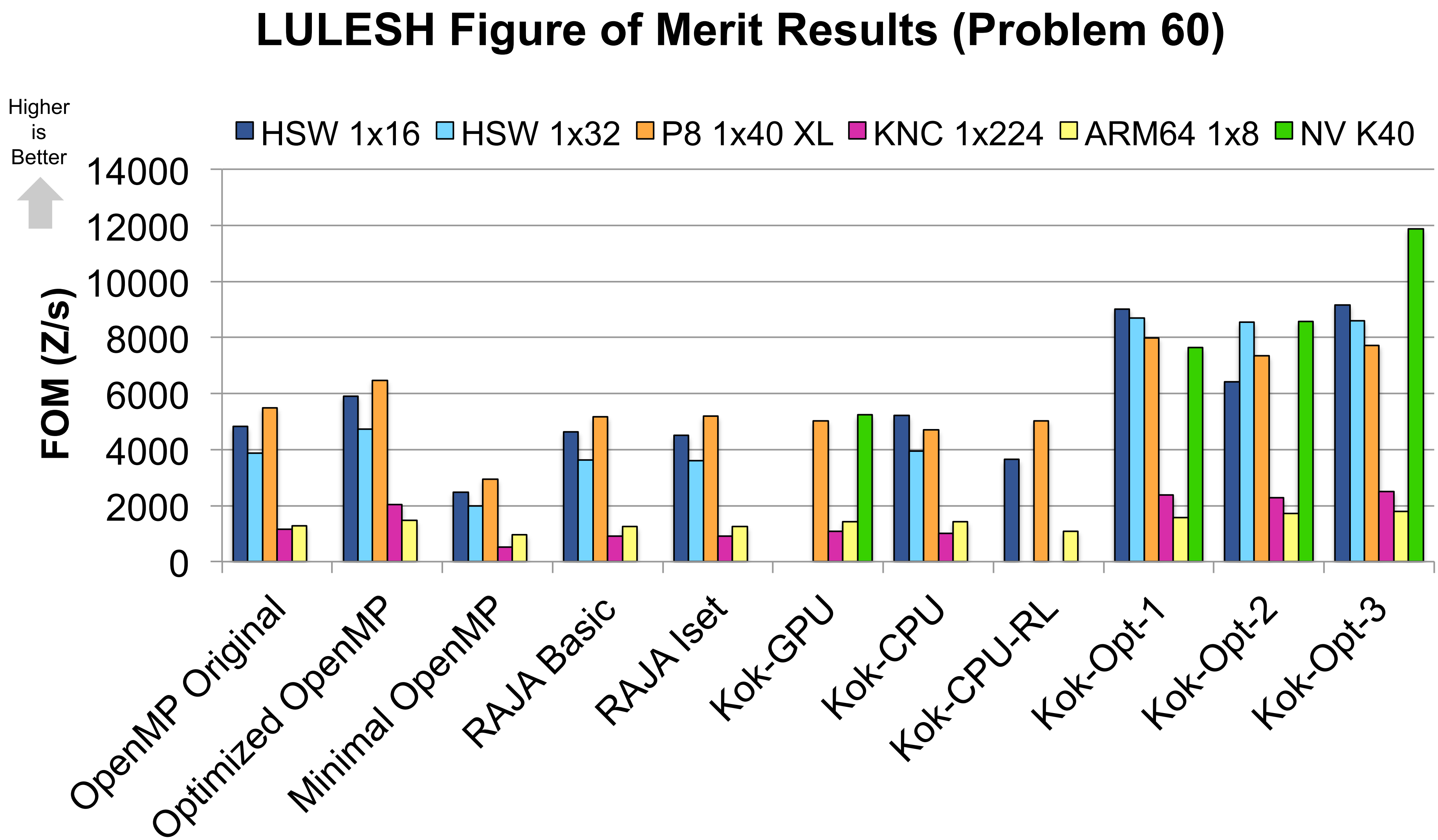
- **Personal opinion – no, in fact, we’re making huge progress but this *is* hard**

- **Internal adoption of the Kokkos Programming Model giving us ability:**
  - Abstract parallel execution dispatch
  - Abstract data access patterns and allocations
  - Retarget code for execution at compile time (including multiple backends in a single application)

- **<u>Proven</u> record of delivering prototypes across multi-core, many-core and GPU devices**
  - POWER8, Xeon Phi (KNC and KNL), Xeon, NVIDIA and recent prototypes on AMD



Kokkos SNL/ASC Study L2 FY15



Results by Dennis Dinge, Christian Trott and Si Hammond

“So Why SoC”?

- **Code abstraction opens up even more opportunities**
  - Much of our mathematics kernels are abstracted (at some level)
  - Particular complex solvers which are key to our application scaling and performance
  - Lots of data structures (meshes) are abstracted at some level
- **Means we can look for opportunities to accelerate our most important kernels with:**
  - Better hardware?
  - More specific fixed-function accelerators (e.g. SoC?)
  - Better software/runtime support
- **Huge potential for impact in performance & energy efficiency**

What Do We Need?

- **Abstractions still need exposure to hardware at the lowest level and are incredibly hard to get right**
  - Can we utilize some of our existing interfaces?
  - System software/runtimes have a huge role to play here
  - Compilers can transform the code for SoC?
  - Mapping to libraries?
- **Want to explore keeping changes to applications to a minimum**
  - Requires us to make decisions about what can accelerate our application *portfolio* the best (Sandia will often answer iterative solvers but there is a more spectrum here)
  - Look for commonalities across our workflow (at SNL and other labs)
  - Leads to mathematics primitives?

GETTING STARTED WITH SOC IN HPC

Fixed function acceleration for basic primitives

Motivating Context

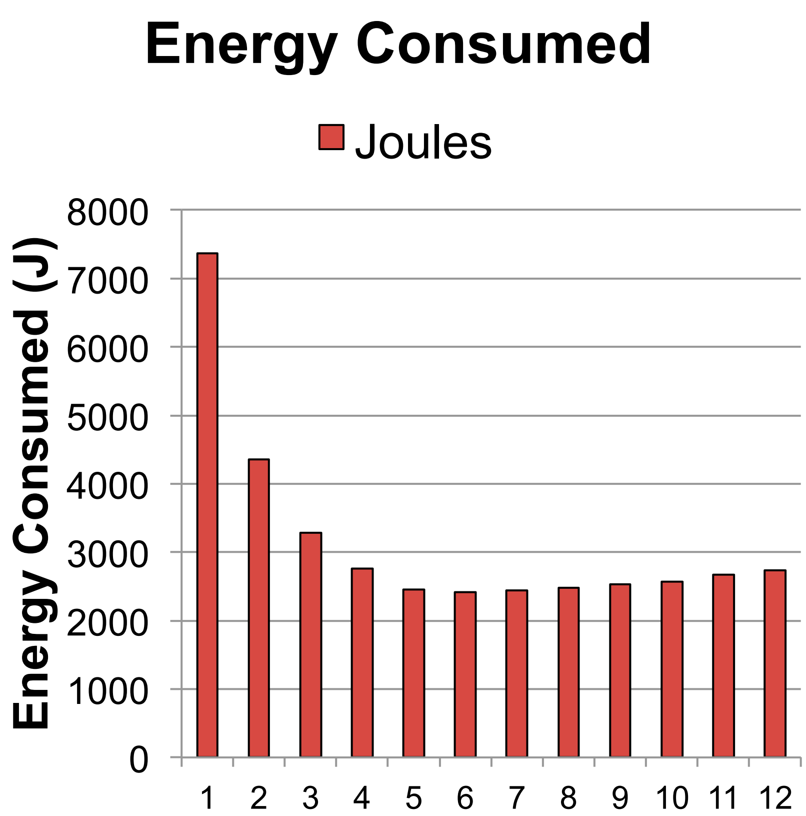
- **Motivating Use Case: MiniFE CG Solve**

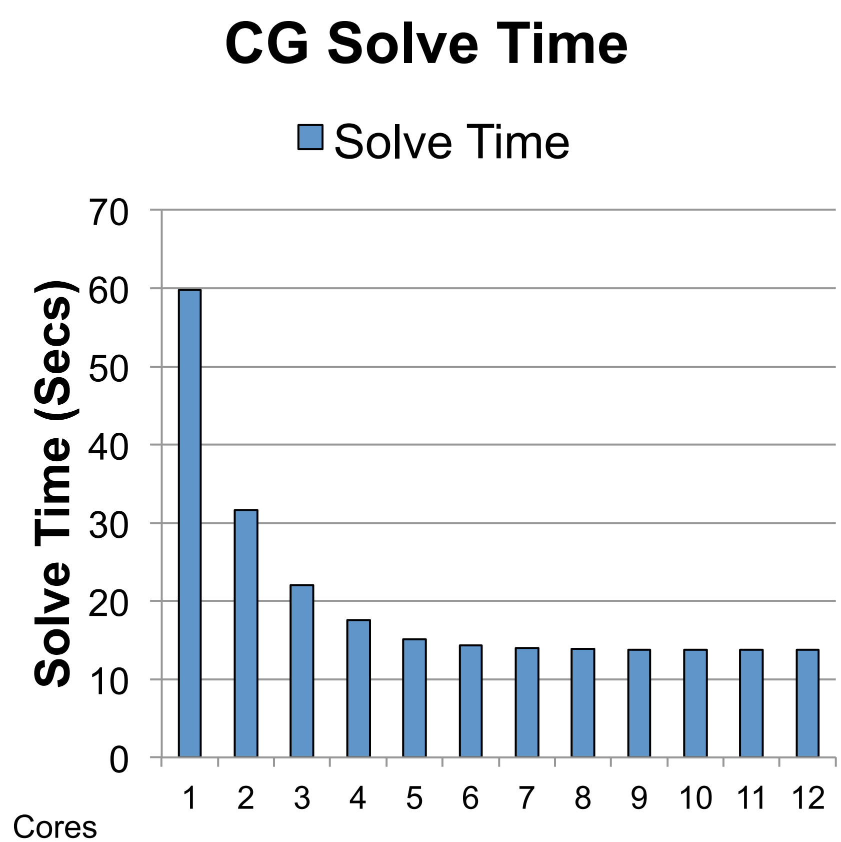
- Simplistic but represents kernels which are important to ASC application portfolio at Sandia
- **Dominated (in time) by Sparse Matrix-Vector Products**
- Heavily memory bound, saturate memory sub-system quickly
- Insufficient balance in processor to meet all the demands of the cores

- Dual-socket Ivy Bridge XC30, 2.4GHz, 12-cores/socket
- Optimized libraries, Intel 15.2.164 compiler, AVX-enabled

MiniFE Simple CG Solve

Sandia National Laboratories

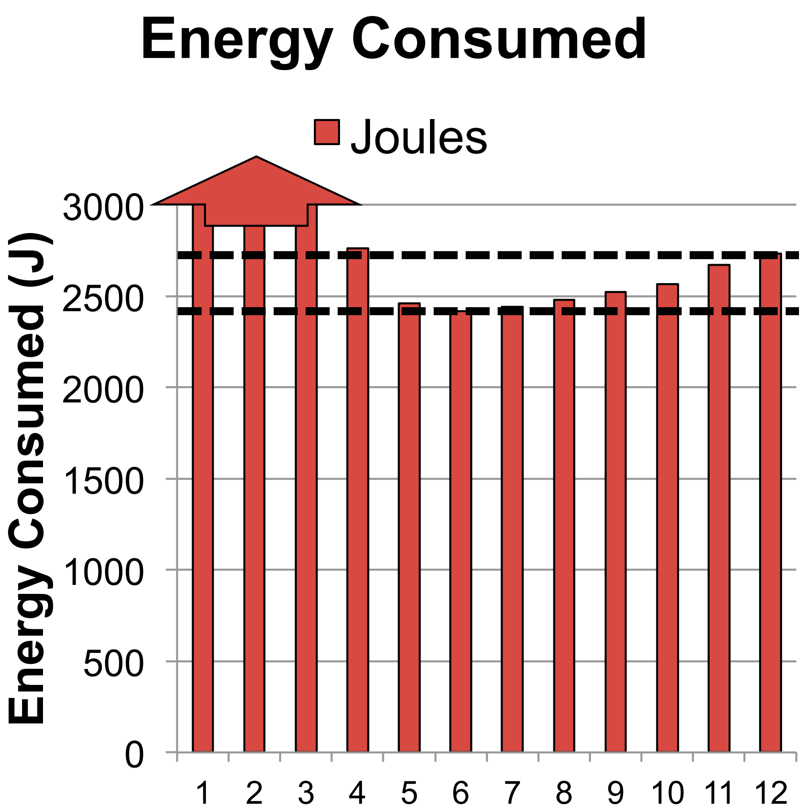


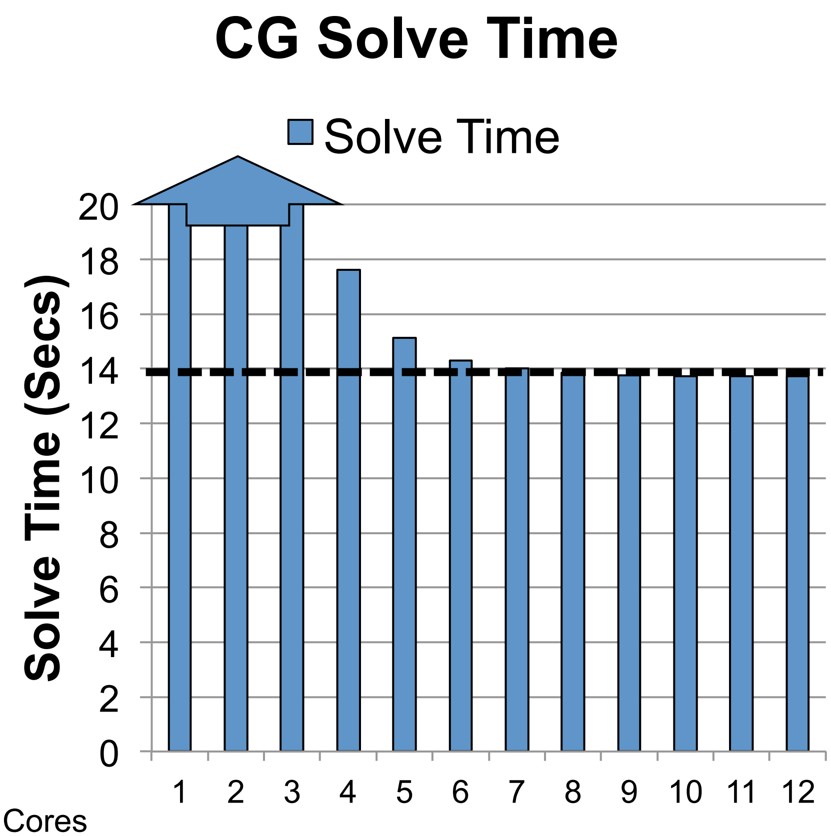


- Simple Finite Element Mesh
- Basic solve, simple kernels, optimized for OpenMP and AVX

Work with S. Hammond, S. Olivier, T.J. Mannos and J. Lewy (Sandia)

Looking Deeper…





- Approximately 10% - 12% increase in energy consumption
- No significant change in runtime

Work with S. Hammond, S. Olivier, T.J. Mannos and J. Lewy (Sandia)

Thoughts on MiniFE Example

- **Small, very simple example – this is just *part* of an application**
  - When combined into larger codes we see many *different* behaviors
  - Input and problem dependent
- Given the importance of some kernels can we make accelerated functions a part of our design?
- Showed small gain in energy efficiency
- Performance wins are less clear in this example which is heavily memory bandwidth bound
  - … but we are busy thinking about this problem

Can We Have Impact?

- The SpMV kernels in MiniFE are the workhouse of *some* of Sandia's workflow (but an important class of problem)

- Between 40 to 95% of application time spent on these kernels in **real** problems

- Scale with memory bandwidth not computational performance
  - So have seen very poor optimization over the past decade

- Seeing similar uses in analytics and commercial environments

DISCUSSION AND THOUGHTS

Summary

Sandia National Laboratories

- **HPC applications are large and complex, even simple ones are hard to rewrite, there are *many* in the community that we depend on**
  - This is going to cost serious dollars and time if we really *make* developers rewrite their code (ASC could be O($M) – O($Bn))
  - Validation and verification costs for climate, weapons etc are *huge* (and in some ways may totally dominate our *real* cost)
  - Need to consider total workflow and not just the “sexy” scientific simulation

- **At some level there really are common kernels and patterns**
  - Think of Phil Colella’s Application Dwarves (still drives how I think about our community)
  - Doesn’t cover 100% of codes but we will <u>never</u> remove the need for general purpose processor cores

On SoC..

- **SoC is an opportunity to rethink our plans for Exascale**
  - Think smaller general purpose, silicon devoted to the things we actually run
  - Non-trivial and pushes complexity to the runtimes, libraries and compilers
  - But this is an area where these communities tend to work best
- **In my opinion we need to focus on areas where data movement limits performance**
  - Move computation to the data (fixed functions?)
  - More efficient mechanisms to handle data movement (gather/scatter)
  - Parallelism enablement which the wider community may not need (particularly complex atomic operations in memory)

Resources

- Many resources we use day-to-day are online or significant parts are online:
  - https://github.com/spdomin/Nalu (Single Physics App)
  - https://github.com/trilinos/trilinos (Solvers)
  - https://github.com/kokkos/kokkos (C++ Programming Model)

  - https://github.com/sstsimulator (HW Simulation Infrastructure)
  - http://www.cs.sandia.gov/qthreads/ (Lightweight On-node Tasking)
  - http://www.cs.sandia.gov/Portals (NIC Acceleration)
  - http://www.mantevo.org (Mini-Apps)

- Continue to look for great summer students, interns, post-docs and staff .. come be part of our team!

Sandia National Laboratories

Exceptional service in the national interest