SAND2015-7627PE

Exceptional service in the national interest

Sandia National Laboratories



Tools for Next Generation Platforms

Si Hammond
Scalable Computer Architectures, Sandia National Labs, NM, USA
sdhammo@sandia.gov

U.S. DEPARTMENT OF ENERGY | NNSA

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.

Talk of Two Halves

Sandia National Laboratories

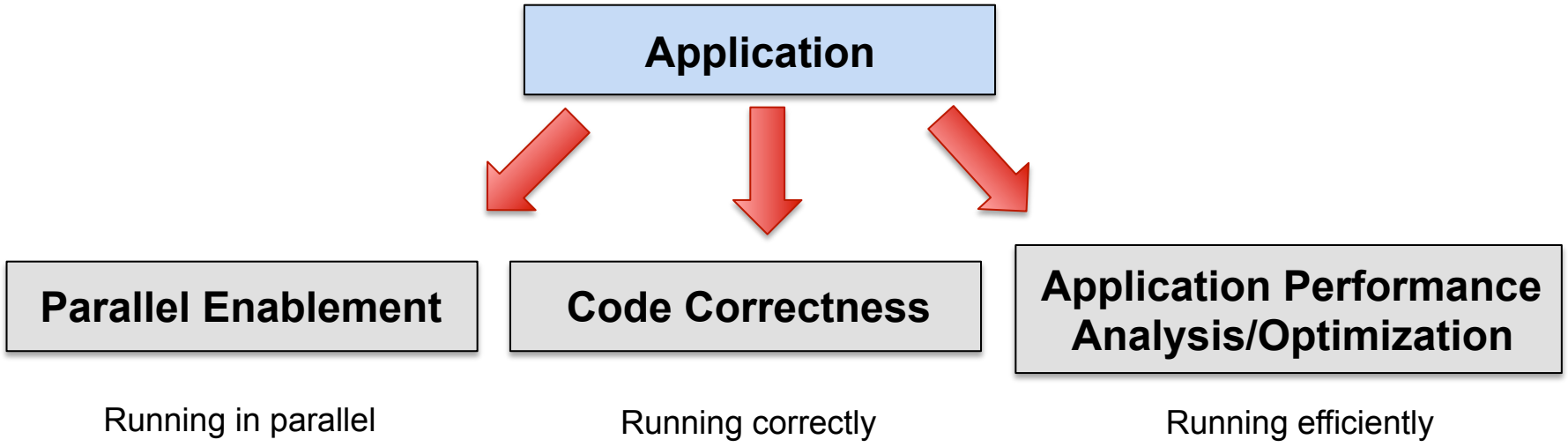
- **Part 1:** What tools do we currently have on the test beds that you can use/what can they be used for?
  - **Parallel Enablement**
  - **Code Correctness**
  - **Performance optimizations**
- **Part 2:** What are we adding to Kokkos to enable lightweight profiling?
  - **KokkosP Interface**
  - **APEX Connector**

Sandia National Laboratories

TOOLS ON THE TEST BEDS

Categories of Tools

Sandia National Laboratories



- Most of these tools can be used throughout application lifetime
- Possibly integrated into overnight builds

Sandia National Laboratories

PARALLEL ENABLEMENT

Parallel Enablement

Sandia National Laboratories

- Intel Advisor XE – test beds (SON and SRN)
  - **Step 1:** Lightweight profile of application including loops
  - **Step 2:** Annotate loops with macros
  - **Step 3:** Advisor generates a prediction of parallel performance
    - Cilk, TBB, OpenMP and pthreads on Xeon and Xeon Phi
  - **Step 4:** Limited correctness check
  - **Step 5:** Convert to parallel model (replace macros)

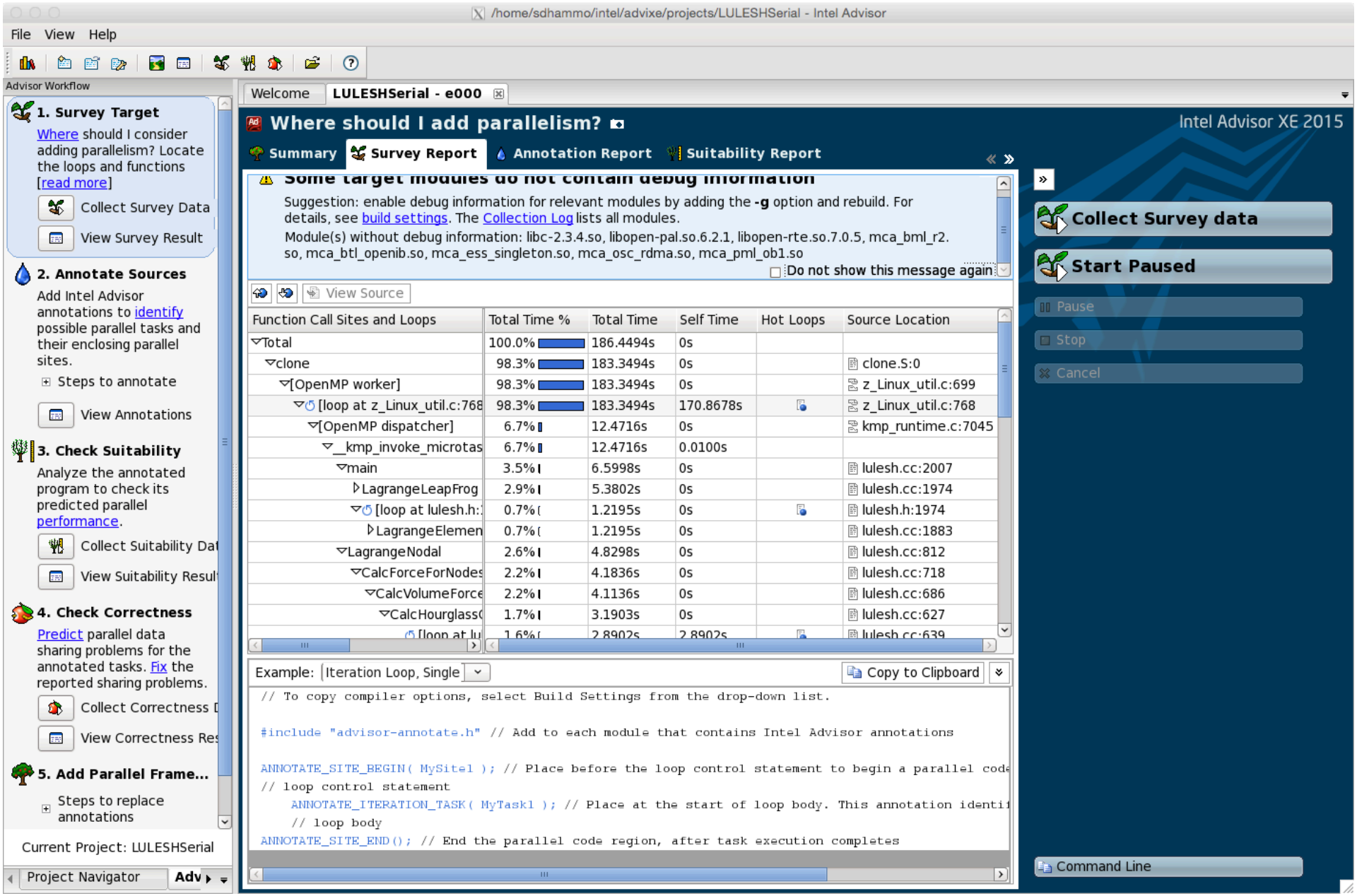
- Trinity will have Cray Reveal Tool (SRN)
  - Requires compilation with the Cray compiler
  - Similar approach, profile based assessment of loops

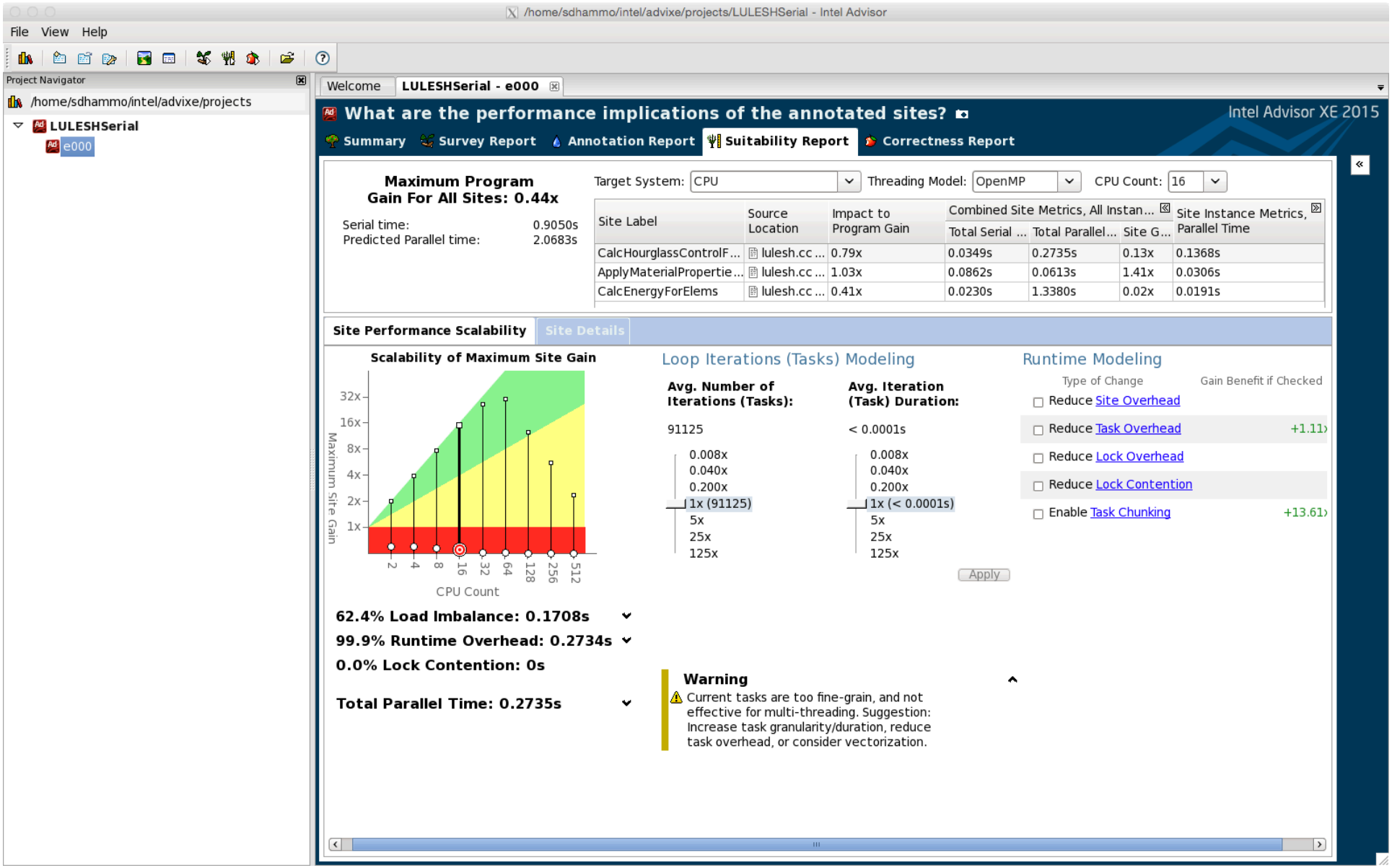
Parallel Enablement

Sandia National Laboratories

- Parallel Advisor XE 2016 will also have initial support for:
  - Vectorization analysis including efficiency metrics
    - AVX128, 256 and 512 (KNL)
  - Loop-based analysis, does not look at block vectorization
  - Initial testing **very** successful
  - Does not require feedback from counters (horray!)

- And… very initial support for:
  - Data structure placement for HBM vs. DDR on the KNL
  - Still a work in progress
  - Assumes data structures will live in the allocation for ever


/home/sdhammo/intel/advixe/projects/LULESHSerial - Intel Advisor
File View Help
Advisor Workflow
1. Survey Target
Where should I consider adding parallelism? Locate the loops and functions [read more]
Collect Survey Data
View Survey Result
2. Annotate Sources
Add Intel Advisor annotations to identify possible parallel tasks and their enclosing parallel sites.
Steps to annotate
View Annotations
3. Check Suitability
Analyze the annotated program to check its predicted parallel performance.
Collect Suitability Dat
View Suitability Result
4. Check Correctness
Predict parallel data sharing problems for the annotated tasks. Fix the reported sharing problems.
Collect Correctness D
View Correctness Res
5. Add Parallel Frame...
Steps to replace annotations
Current Project: LULESHSerial
Project Navigator
Adv
Welcome
LULESHSerial - e000
Where should I add parallelism?
Intel Advisor XE 2015
Summary
Survey Report
Annotation Report
Suitability Report
Some target modules do not contain debug information
Suggestion: enable debug information for relevant modules by adding the -g option and rebuild. For details, see build settings. The Collection Log lists all modules.
Module(s) without debug information: libc-2.3.4.so, libopen-pal.so.6.2.1, libopen-rte.so.7.0.5, mca_bml_r2.so, mca_btl_openib.so, mca_ess_singleton.so, mca_osc_rdma.so, mca_pml_ob1.so
Do not show this message again
View Source
Function Call Sites and Loops | Total Time % | Total Time | Self Time | Hot Loops | Source Location
Total | 100.0% | 186.4494s | 0s
clone | 98.3% | 183.3494s | 0s | clone.S:0
[OpenMP worker] | 98.3% | 183.3494s | 0s | z_Linux_util.c:699
[loop at z_Linux_util.c:768 | 98.3% | 183.3494s | 170.8678s | z_Linux_util.c:768
[OpenMP dispatcher] | 6.7% | 12.4716s | 0s | kmp_runtime.c:7045
__kmp_invoke_microtas | 6.7% | 12.4716s | 0.0100s
main | 3.5% | 6.5998s | 0s | lulesh.cc:2007
LagrangeLeapFrog | 2.9% | 5.3802s | 0s | lulesh.cc:1974
[loop at lulesh.h: | 0.7% | 1.2195s | 0s | lulesh.h:1974
LagrangeElemen | 0.7% | 1.2195s | 0s | lulesh.cc:1883
LagrangeNodal | 2.6% | 4.8298s | 0s | lulesh.cc:812
CalcForceForNodes | 2.2% | 4.1836s | 0s | lulesh.cc:718
CalcVolumeForce | 2.2% | 4.1136s | 0s | lulesh.cc:686
CalcHourglass | 1.7% | 3.1903s | 0s | lulesh.cc:627
[loop at lu | 1.6% | 2.8902s | 2.8902s | lulesh.cc:639
Example: Iteration Loop, Single
Copy to Clipboard
// To copy compiler options, select Build Settings from the drop-down list.
#include "advisor-annotate.h" // Add to each module that contains Intel Advisor annotations
ANNOTATE_SITE_BEGIN(MySite1); // Place before the loop control statement to begin a parallel code
// loop control statement
ANNOTATE_ITERATION_TASK(MyTask1); // Place at the start of loop body. This annotation identi
// loop body
ANNOTATE_SITE_END(); // End the parallel code region, after task execution completes
Collect Survey data
Start Paused
Pause
Stop
Cancel
Command Line


/home/sdhammo/intel/advixe/projects/LULESHSerial - Intel Advisor
File View Help
Project Navigator
/home/sdhammo/intel/advixe/projects
LULESHSerial
e000
Welcome
LULESHSerial - e000
What are the performance implications of the annotated sites?
Intel Advisor XE 2015
Summary
Survey Report
Annotation Report
Suitability Report
Correctness Report
Maximum Program Gain For All Sites: 0.44x
Serial time: 0.9050s
Predicted Parallel time: 2.0683s
Target System: CPU
Threading Model: OpenMP
CPU Count: 16
Site Label
Source Location
Impact to Program Gain
Combined Site Metrics, All Instan...
Total Serial ...
Total Parallel...
Site G...
Site Instance Metrics, Parallel Time
CalcHourglassControlF... lulesh.cc ... 0.79x 0.0349s 0.2735s 0.13x 0.1368s
ApplyMaterialPropertie... lulesh.cc ... 1.03x 0.0862s 0.0613s 1.41x 0.0306s
CalcEnergyForElems lulesh.cc ... 0.41x 0.0230s 1.3380s 0.02x 0.0191s
Site Performance Scalability
Site Details
Scalability of Maximum Site Gain
Maximum Site Gain
32x
16x
8x
4x
2x
1x
2
4
8
16
32
64
128
256
512
CPU Count
62.4% Load Imbalance: 0.1708s
99.9% Runtime Overhead: 0.2734s
0.0% Lock Contention: 0s
Total Parallel Time: 0.2735s
Loop Iterations (Tasks) Modeling
Avg. Number of Iterations (Tasks):
91125
0.008x
0.040x
0.200x
1x (91125)
5x
25x
125x
Avg. Iteration (Task) Duration:
< 0.0001s
0.008x
0.040x
0.200x
1x (< 0.0001s)
5x
25x
125x
Apply
Runtime Modeling
Type of Change
Gain Benefit if Checked
Reduce Site Overhead
Reduce Task Overhead +1.11x
Reduce Lock Overhead
Reduce Lock Contention
Enable Task Chunking +13.61x
Warning
Current tasks are too fine-grain, and not effective for multi-threading. Suggestion: Increase task granularity/duration, reduce task overhead, or consider vectorization.

Notes

Sandia National Laboratories

- Advisor XE is loop based which maps well to OpenMP and Kokkos
- Does not “see” the modifications needed for algorithm refactoring
- Assumes data structures are not going to be changed to enable vectorization or parallelism
- Likely not to get the performance than a well tuned algorithm by hand will but very much quicker
- Good idea is to set the performance model to a large number of threads (e.g. 256) to catch low-level serialization overheads

Sandia National Laboratories

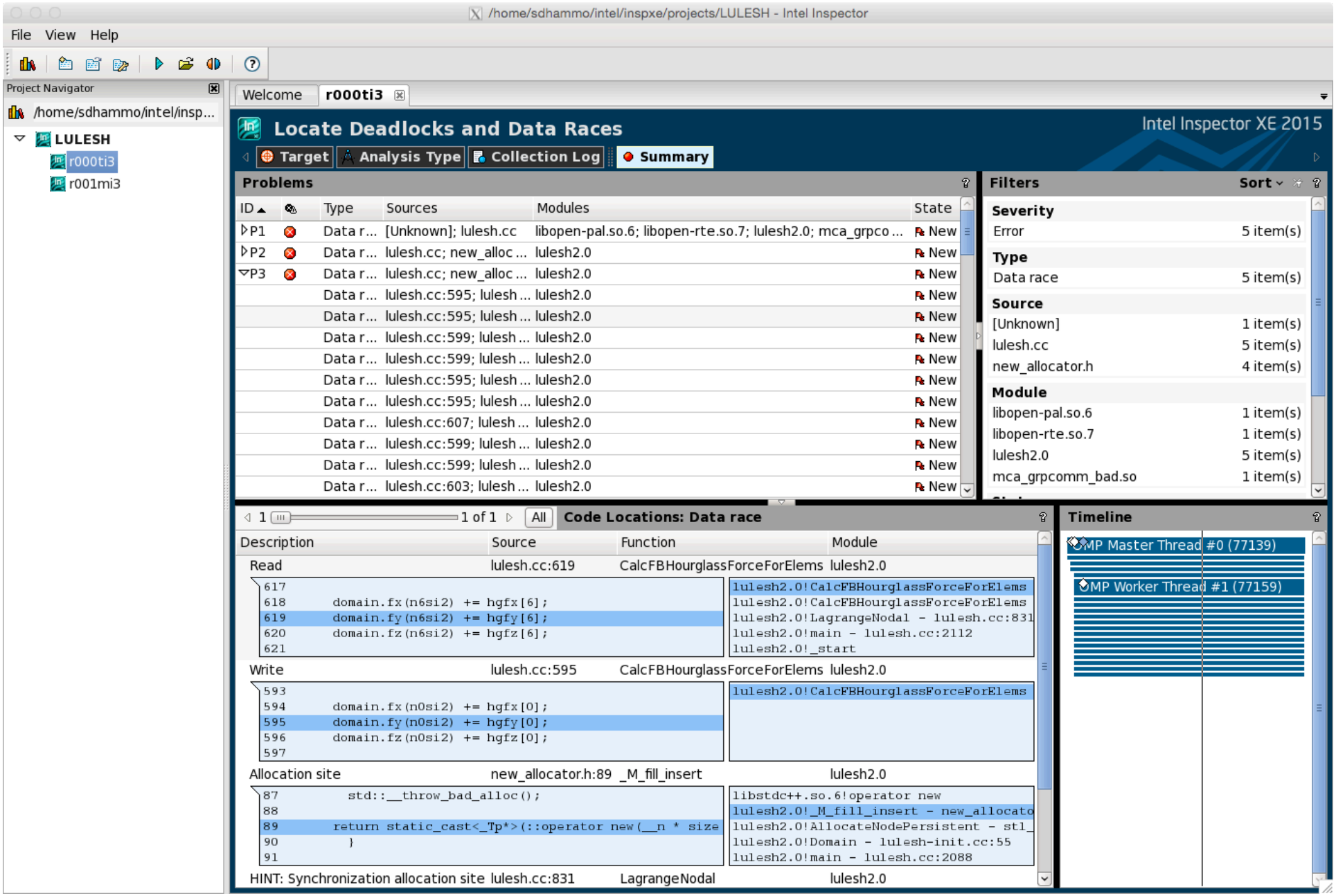
CODE CORRECTNESS

Code Correctness

Sandia National Laboratories

- Wider range of tools than parallel enablement reflecting longer history

- Valgrind
- Compilation based sanitizers (clang and GNU >= 4.9)
  - ThreadSanitizer (thread race conditions)
  - AddressSanitizer (buffer overflows and use-after-free)
  - MemorySanitizer (finds reads from uninitialized memory)
- Intel Inspector XE
  - Memory leak detection
  - Thread race conditions


/home/sdhammo/intel/inspxe/projects/LULESH - Intel Inspector
File View Help
Project Navigator
/home/sdhammo/intel/insp...
LULESH
r000ti3
r001mi3
Welcome
r000ti3
Locate Deadlocks and Data Races
Intel Inspector XE 2015
Target
Analysis Type
Collection Log
Summary
Problems
ID Type Sources Modules State
P1 Data r... [Unknown]; lulesh.cc libopen-pal.so.6; libopen-rte.so.7; lulesh2.0; mca_grpco ... New
P2 Data r... lulesh.cc; new_alloc ... lulesh2.0 New
P3 Data r... lulesh.cc; new_alloc ... lulesh2.0 New
Data r... lulesh.cc:595; lulesh ... lulesh2.0 New
Data r... lulesh.cc:595; lulesh ... lulesh2.0 New
Data r... lulesh.cc:599; lulesh ... lulesh2.0 New
Data r... lulesh.cc:599; lulesh ... lulesh2.0 New
Data r... lulesh.cc:595; lulesh ... lulesh2.0 New
Data r... lulesh.cc:595; lulesh ... lulesh2.0 New
Data r... lulesh.cc:607; lulesh ... lulesh2.0 New
Data r... lulesh.cc:599; lulesh ... lulesh2.0 New
Data r... lulesh.cc:599; lulesh ... lulesh2.0 New
Data r... lulesh.cc:603; lulesh ... lulesh2.0 New
Filters
Sort
Severity
Error 5 item(s)
Type
Data race 5 item(s)
Source
[Unknown] 1 item(s)
lulesh.cc 5 item(s)
new_allocator.h 4 item(s)
Module
libopen-pal.so.6 1 item(s)
libopen-rte.so.7 1 item(s)
lulesh2.0 5 item(s)
mca_grpcomm_bad.so 1 item(s)
1 1 of 1 All
Code Locations: Data race
Description Source Function Module
Read lulesh.cc:619 CalcFBHourglassForceForElems lulesh2.0
617
618 domain.fx(n6si2) += hgfx[6];
619 domain.fy(n6si2) += hgfy[6];
620 domain.fz(n6si2) += hgfz[6];
621
lulesh2.0!CalcFBHourglassForceForElems
lulesh2.0!CalcFBHourglassForceForElems
lulesh2.0!LagrangeNodal - lulesh.cc:831
lulesh2.0!main - lulesh.cc:2112
lulesh2.0!_start
Write lulesh.cc:595 CalcFBHourglassForceForElems lulesh2.0
593
594 domain.fx(n0si2) += hgfx[0];
595 domain.fy(n0si2) += hgfy[0];
596 domain.fz(n0si2) += hgfz[0];
597
lulesh2.0!CalcFBHourglassForceForElems
Allocation site new_allocator.h:89 _M_fill_insert lulesh2.0
87 std::__throw_bad_alloc();
88
89 return static_cast<_Tp*>(::operator new(__n * size
90 }
91
libstdc++.so.6!operator new
lulesh2.0!_M_fill_insert - new_allocato
lulesh2.0!AllocateNodePersistent - stl_
lulesh2.0!Domain - lulesh-init.cc:55
lulesh2.0!main - lulesh.cc:2088
HINT: Synchronization allocation site lulesh.cc:831 LagrangeNodal lulesh2.0
Timeline
OMP Master Thread #0 (77139)
OMP Worker Thread #1 (77159)

Notes

Sandia National Laboratories

- Compilation based sanitizers slow code down a lot (>8X)
  - And usually just fatal when the issue is detected (not always)
- Inspector XE will be able to catch multiple errors in a single run
  - Get a long list presented back to the user
  - Can be very slow to perform full thread memory analysis (> 20X)
  - Not a perfect science, based on heuristics
  - Very good for memory leak detection
- Valgrind still works on all the test beds

MPI Correctness

Sandia National Laboratories

- Intel's new MPI 5.0 has an MPI correctness tracer
  - Step 1: Trace MPI events using options on the MPI runtime
  - Step 2: Use the Trace Analyzer for a correctness check
- Can check for errors:
  - Data buffers provided to MPI
  - Data type matching across calls
  - Some message deadlock situations (even when the application doesn't complete correctly)
  - Overlapping buffers
- Scales reasonably well, Intel testing up to 64K cores

Sandia National Laboratories

PERFORMANCE ANALYSIS/OPTIMIZATION

Performance Analysis

Sandia National Laboratories

- Lots of options here but not all map to platforms well

- Intel Vtune Amplifier XE (Xeon, Xeon Phi)
- Cray PAT (Xeon and will be on Trinity Phase-II KNL)
- NVIDIA Nsight (NVIDIA GPUs)
- AMD CodeXL (AMD Fusion APUs)

- TAU
- OpenSpeedShop

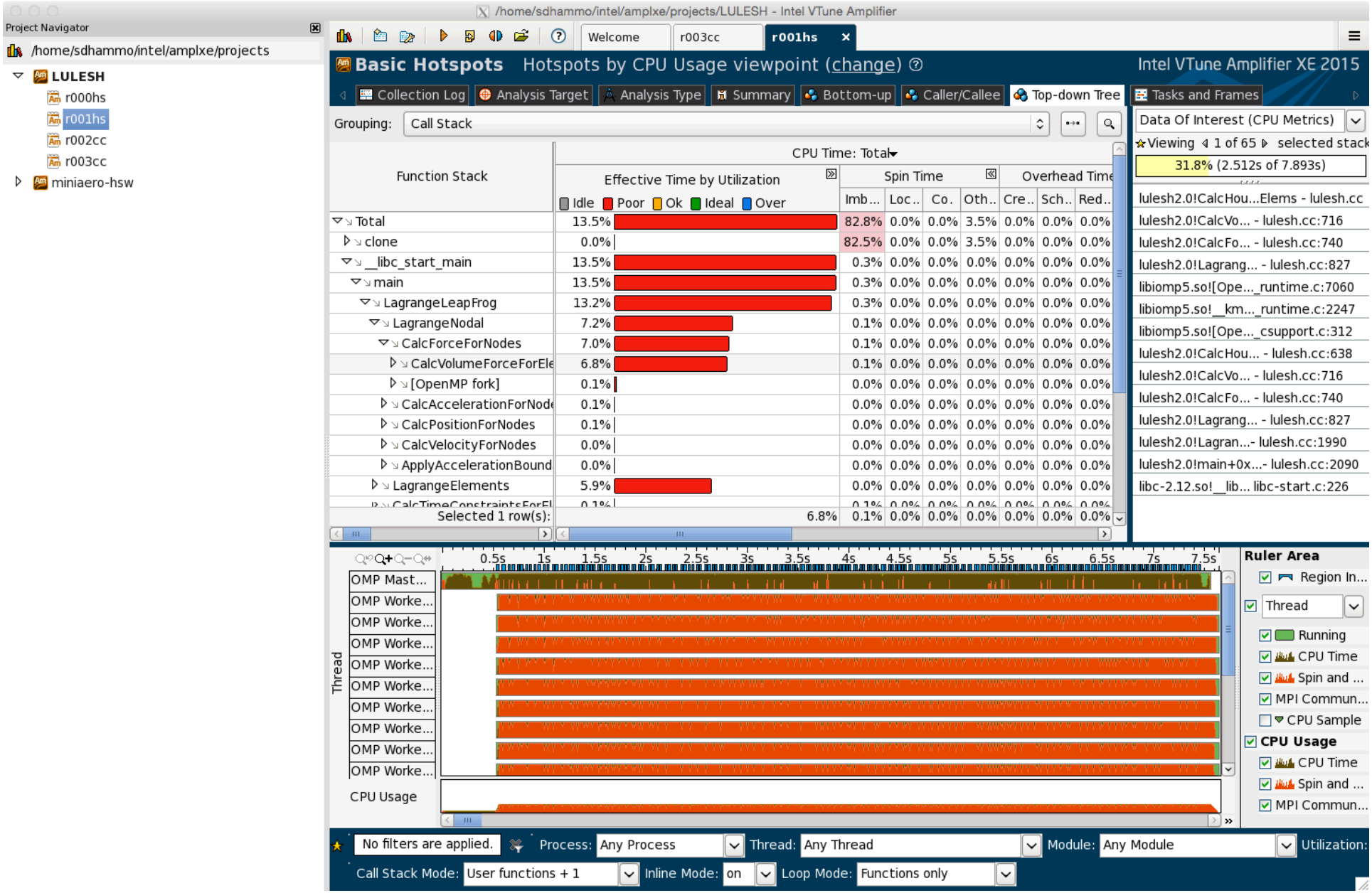
- On CORAL will be HPCToolkit (Rice Univ.)

Capabilities

Sandia National Laboratories

Tool	Profile MPI Jobs	OpenMP	GPU	Perf. Count
VTune XE	Not really	Yes	Yes (Intel)	Yes
Cray PAT	Yes	Yes	No	Yes
NSight	No	No	Yes (NV)	Yes (GPU)
CodeXL	No	No	Yes (AMD)	Yes (APU)
TAU	Yes	Yes	Yes (NV)	Yes
OpenSS	Yes	Yes	No	Yes
HPC Toolkit	Yes	Yes	Yes (NV)	Yes

- Big variation in usability across tools and built in analysis


/home/sdhammo/intel/amplxe/projects/LULESH - Intel VTune Amplifier
Project Navigator
/home/sdhammo/intel/amplxe/projects
LULESH
r000hs
r001hs
r002cc
r003cc
miniaero-hsw
Welcome
r003cc
r001hs
Basic Hotspots
Hotspots by CPU Usage viewpoint (change)
Intel VTune Amplifier XE 2015
Collection Log
Analysis Target
Analysis Type
Summary
Bottom-up
Caller/Callee
Top-down Tree
Tasks and Frames
Grouping:
Call Stack
Data Of Interest (CPU Metrics)
Viewing 1 of 65 selected stack
31.8% (2.512s of 7.893s)
Function Stack
CPU Time: Total
Effective Time by Utilization
Idle Poor Ok Ideal Over
Spin Time
Overhead Time
Imb...
Loc..
Co.
Oth..
Cre..
Sch..
Red..
Total 13.5% 82.8% 0.0% 0.0% 3.5% 0.0% 0.0% 0.0%
clone 0.0% 82.5% 0.0% 0.0% 3.5% 0.0% 0.0% 0.0%
__libc_start_main 13.5% 0.3% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
main 13.5% 0.3% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
LagrangeLeapFrog 13.2% 0.3% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
LagrangeNodal 7.2% 0.1% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
CalcForceForNodes 7.0% 0.1% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
CalcVolumeForEle 6.8% 0.1% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
[OpenMP fork] 0.1% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
CalcAccelerationForNode 0.1% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
CalcPositionForNodes 0.1% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
CalcVelocityForNodes 0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
ApplyAccelerationBound 0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
LagrangeElements 5.9% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
Selected 1 row(s): 6.8% 0.1% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
lulesh2.0!CalcHou...Elems - lulesh.cc
lulesh2.0!CalcVo... - lulesh.cc:716
lulesh2.0!CalcFo... - lulesh.cc:740
lulesh2.0!Lagrang... - lulesh.cc:827
libiomp5.so![Ope..._runtime.c:7060
libiomp5.so!__km..._runtime.c:2247
libiomp5.so![Ope..._csupport.c:312
lulesh2.0!CalcHou... - lulesh.cc:638
lulesh2.0!CalcVo... - lulesh.cc:716
lulesh2.0!CalcFo... - lulesh.cc:740
lulesh2.0!Lagrang... - lulesh.cc:827
lulesh2.0!Lagran...- lulesh.cc:1990
lulesh2.0!main+0x...- lulesh.cc:2090
libc-2.12.so!__lib... libc-start.c:226
0.5s 1s 1.5s 2s 2.5s 3s 3.5s 4s 4.5s 5s 5.5s 6s 6.5s 7s 7.5s
Thread
OMP Mast...
OMP Worke...
CPU Usage
Ruler Area
Region In...
Thread
Running
CPU Time
Spin and ...
MPI Commun...
CPU Sample
CPU Usage
CPU Time
Spin and ...
MPI Commun...
No filters are applied.
Process: Any Process
Thread: Any Thread
Module: Any Module
Utilization:
Call Stack Mode: User functions + 1
Inline Mode: on
Loop Mode: Functions only

What can I do with VTune?

Sandia National Laboratories

- In VTune 2015 you can:
  - Thread scalability analysis “concurrency”
  - Memory bandwidth
  - General Exploration (Why isn’t my application faster?)
  - Hotspot analysis
- In VTune 2016 we expect to have:
  - Better support for Haswell (actually any support for Haswell!)
    - Including QPI counter analysis for multi-socket systems
  - Initial support for KNL models
  - Possibly high-bandwidth memory access analysis
  - Initial energy profiling analysis (very early)

Summary

Sandia National Laboratories

- **I want to work out where to add parallelism or vectorization to my application**
  - Use Parallel Enablement tools (Advisor XE for vectorization)
- **I want to work out if my code is thread safe and executes correctly?**
  - Use Code Correctness tools (Inspector XE for heavyweight thread safety)
- **I want to work out where bottlenecks/performance issues are with my code**
  - Use Performance Optimizations tools (Vtune/Nsight/Cray PAT)

Sandia National Laboratories

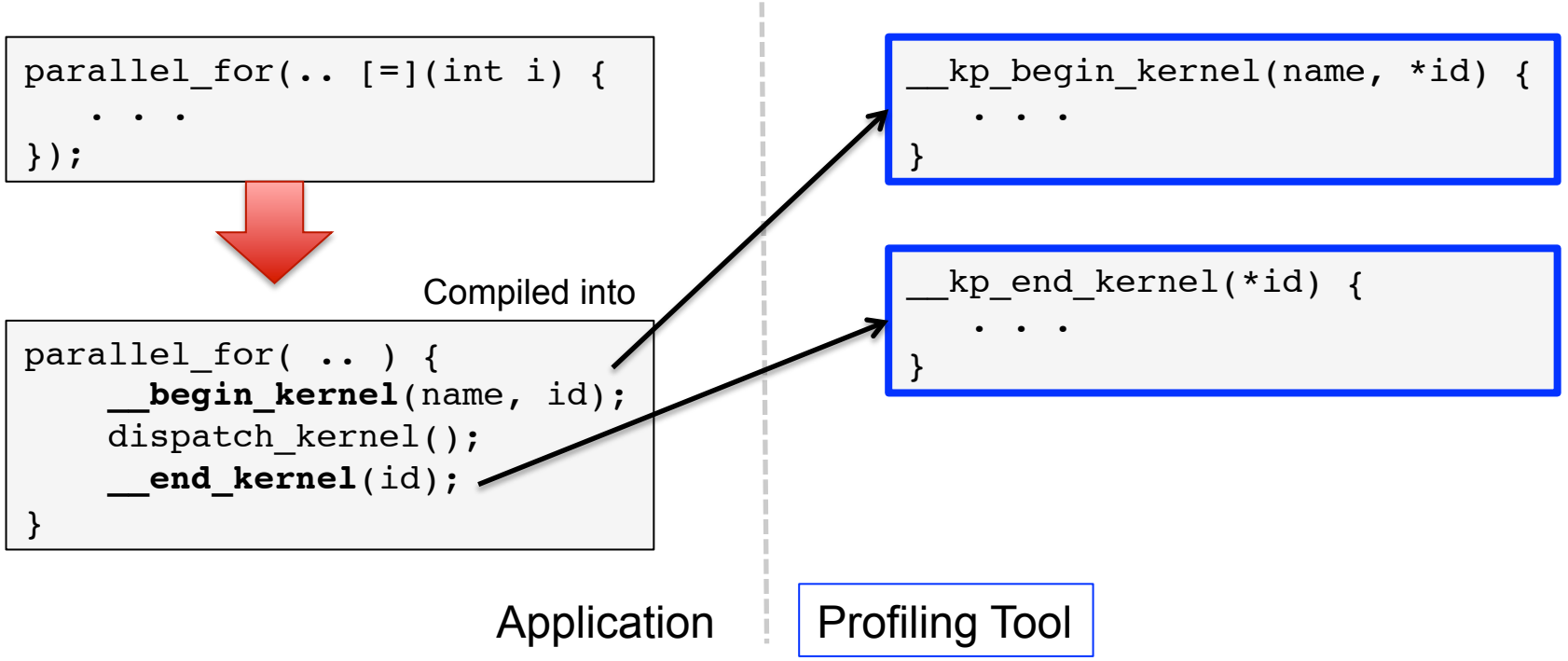
KokkosP and APEX

Profiling Kokkos

Sandia National Laboratories

- Profiling Kokkos is actually quite difficult because of what users without the abstraction expect
  - CUDA seems to be OK because Kokkos behaves as CUDA is normally compiled
  - Directives are a problem (OpenMP, same problem would exist for OpenACC)
- Understanding how Kokkos programs are constructed means we can exploit context (parallel-for etc)
- Want to be able to utilize cross platform tools as much as possible but allow vendors to link into the abstractions as needed

KokkosP Connector



- KOKKOS_PROFILE_LIBRARY="<path to library 1>:<path to library 2> .."
- Full dynamic loading at runtime
- Parallel For, Reduce and Scan

KokkosP Connector

Sandia National Laboratories

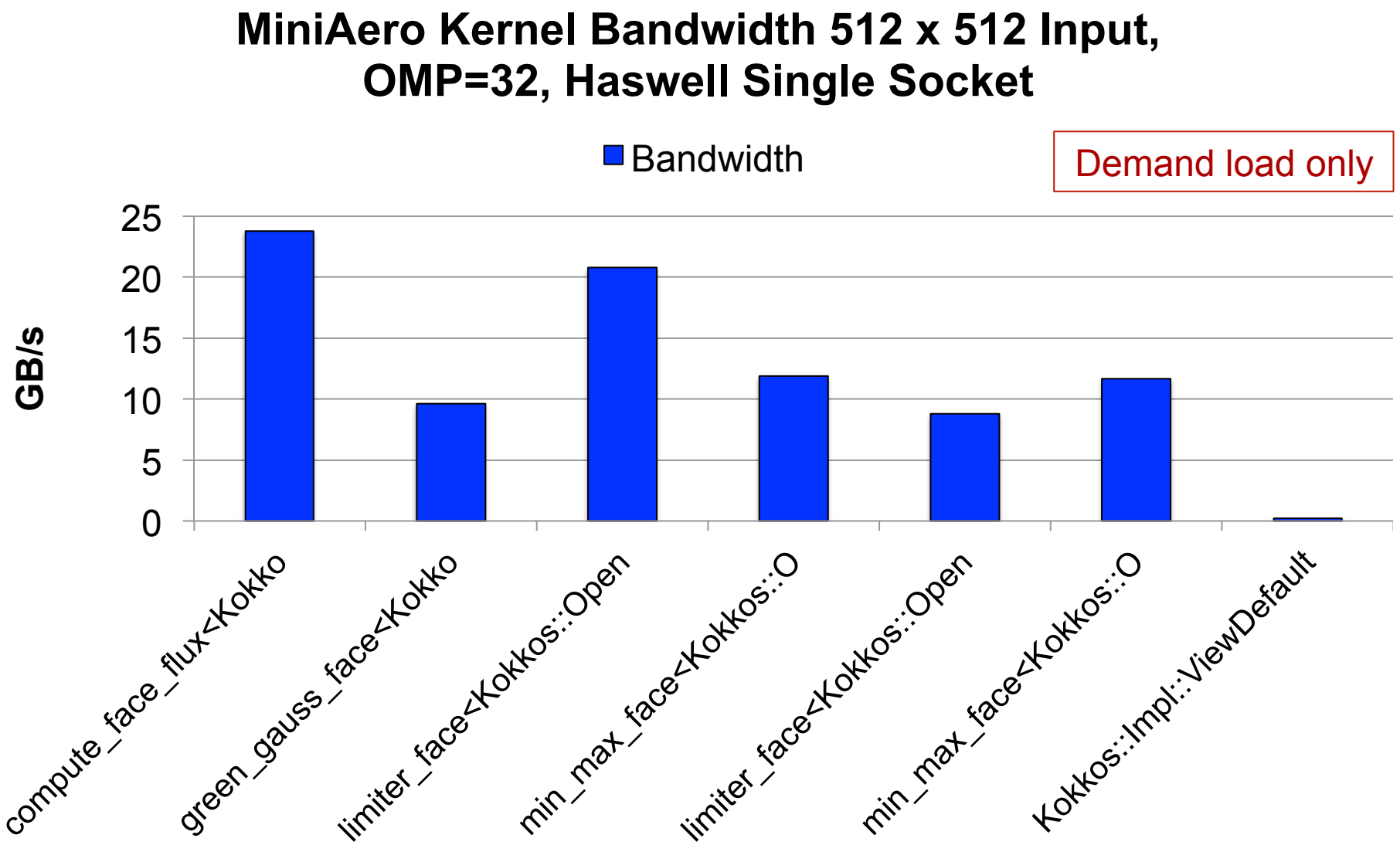
- Currently in testing but is part of the Kokkos master
- To enable: -DKOKKOS_ENABLE_PROFILING
  - Builds hooks into your application but does not build any profiling tools
  - Overhead to have calls in your code with tools not loaded is very low, appears to be negligible in initial timing
  - If you name kernels you will get profiling output relating to that otherwise we use type names to generate a name
- Hoping to make this a default for Kokkos programs but needs to be tested in the community
- "Always" available profiling and logging, no recompile

Initial Performance Analysis Tools

Sandia National Laboratories

- Simple kernel timing profiler (CPU, GPU)
  - Aggregates time spent in individual kernels, time per call, can provide variance to look out for potential load imbalance issues
  - Very lightweight
  - Works for Xeon, Xeon Phi, POWER8 and for NVIDA GPU
- Kernel memory bandwidth profiler on Haswell (CPU only)
  - Tracks demand memory requests made to the DRAM by kernel
  - Total memory read (even writes are reads!) over time
  - Aggregates over total run
- High watermark memory consumption of Kokkos application

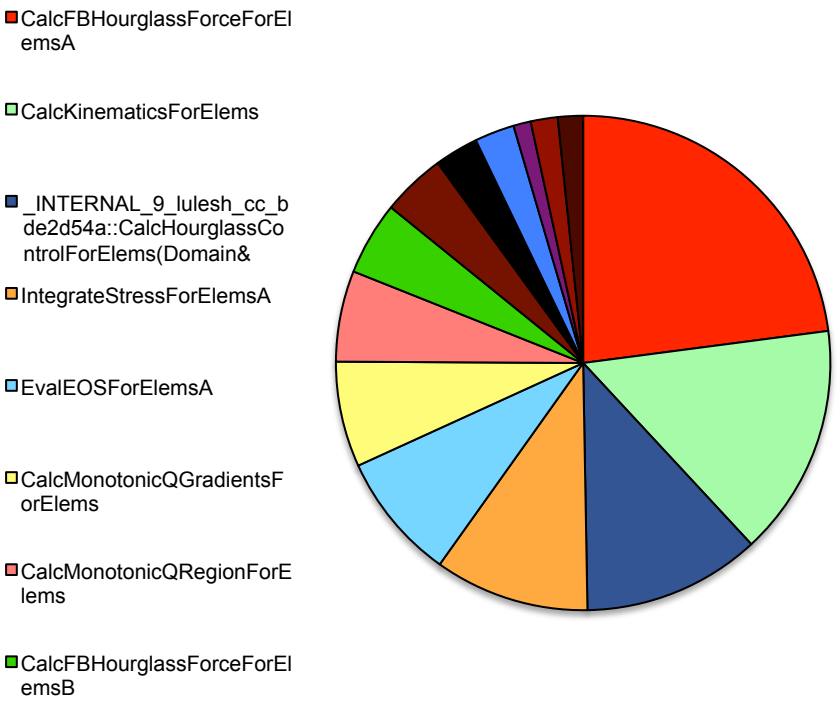
Kernel Memory Bandwidth



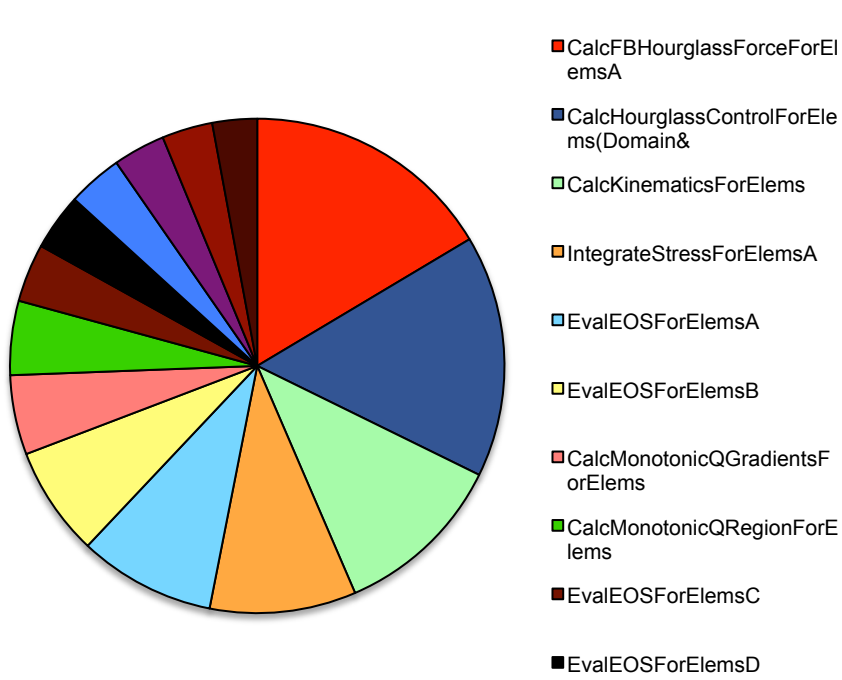
KokkosP Kernel Comparison (LULESH)

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

Prototype Tools

Sandia National Laboratories

- Kokkos OpenMP affinity check (CPU OpenMP backend)
  - Will check that application does not have more threads than provided in the CPU mask
  - Prevent performance/benchmarking errors when using MPI
- Floating Point calculation by kernel (CPU, probably GPU)
  - Counts hardware FLOP/s by kernel
  - But possible to use APEX connector to perform heavyweight vectorization analysis by kernel
  - Will load at bytes loaded to calculation operations ratio

Prototype Prototype Tools

Sandia National Laboratories

- Memory Load Imbalance Detector (CPU)
  - We aggregate most counts by individual thread already
  - So will be able to look at memory subsystem pressure by thread
  - Use this analysis to detect kernel load imbalance

Sandia National Laboratories

APEX – Application Characterization

Performance Characterization

Sandia National Laboratories

- Counters for basic machine operations are almost impossible to obtain accurately and predictably across platforms

- Why?
  - Different chip designers think of events in different ways
  - Performance counters are for validation not for counting operations
  - Different chips behave in different ways, not everyone implements operations the same way
  - Different ISAs, micro-operations *etc etc*

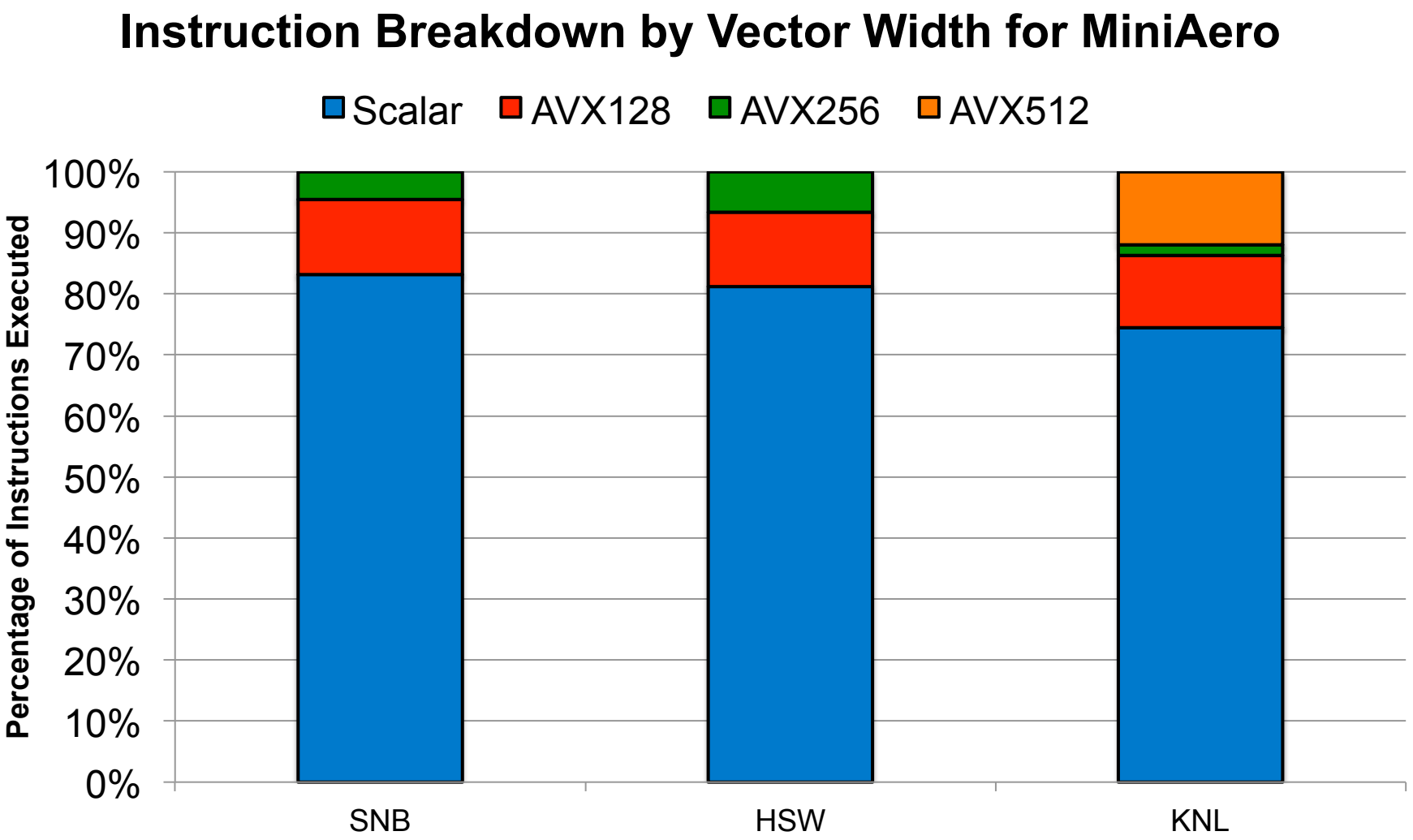
- **Conclusion:** may need to have the ability to accurately count events in a deterministic way

APEX

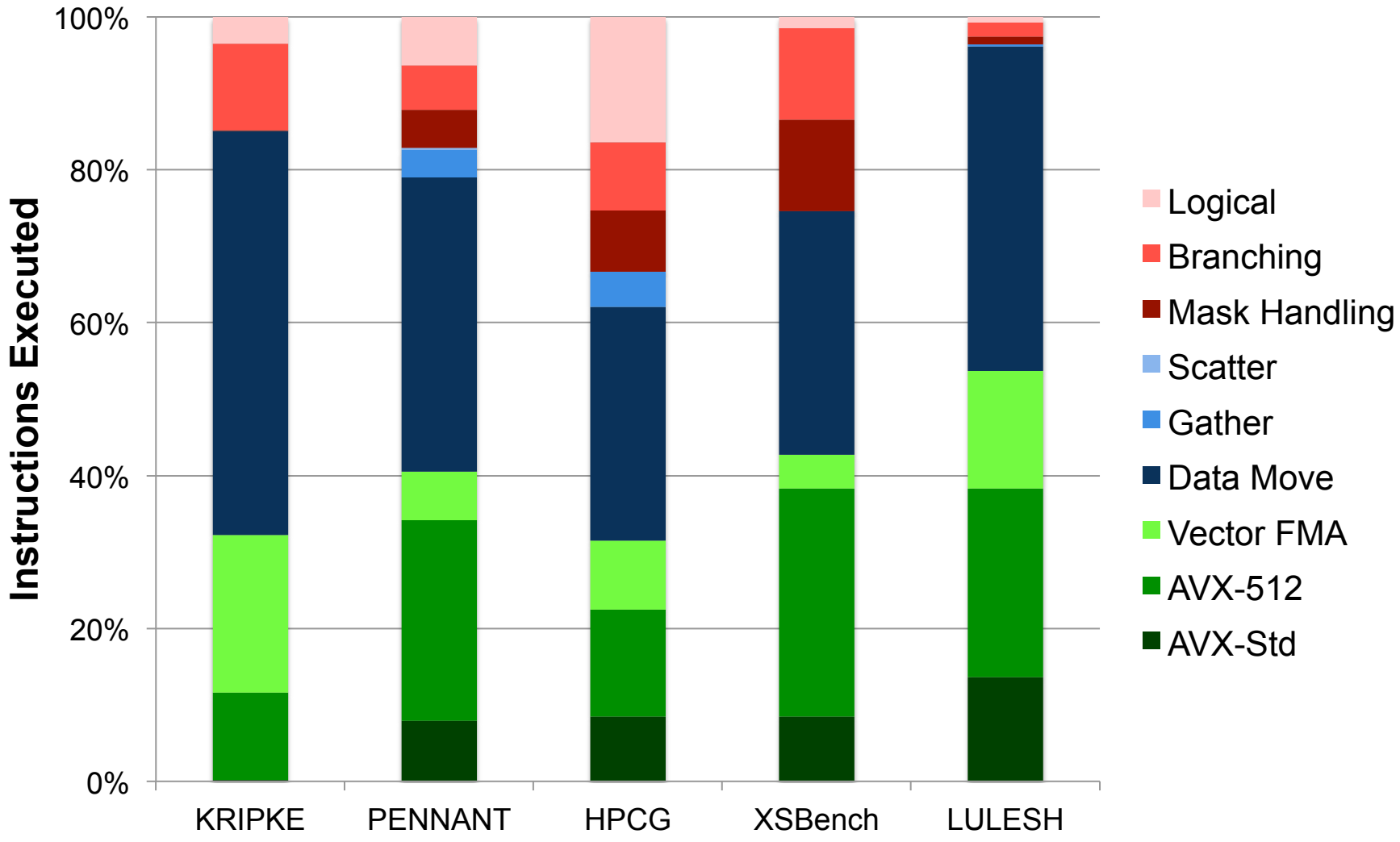
Sandia National Laboratories

- APEX (Application Characterization for Exascale) is a Sandia LDRD project to look at whether these tools can be made
  - Xeon and Xeon Phi based to support problem diagnosis on KNL
- Focus:
  - Vectorization analysis
  - Floating point operations vs. vectored logicals, conditionals etc
  - Memory load/store behavior
  - Gather/Scatter analysis
  - Eventually masked vector operation analysis
- Mostly complete but user experience is very poor

Emulation and Instruction Analysis for KNL

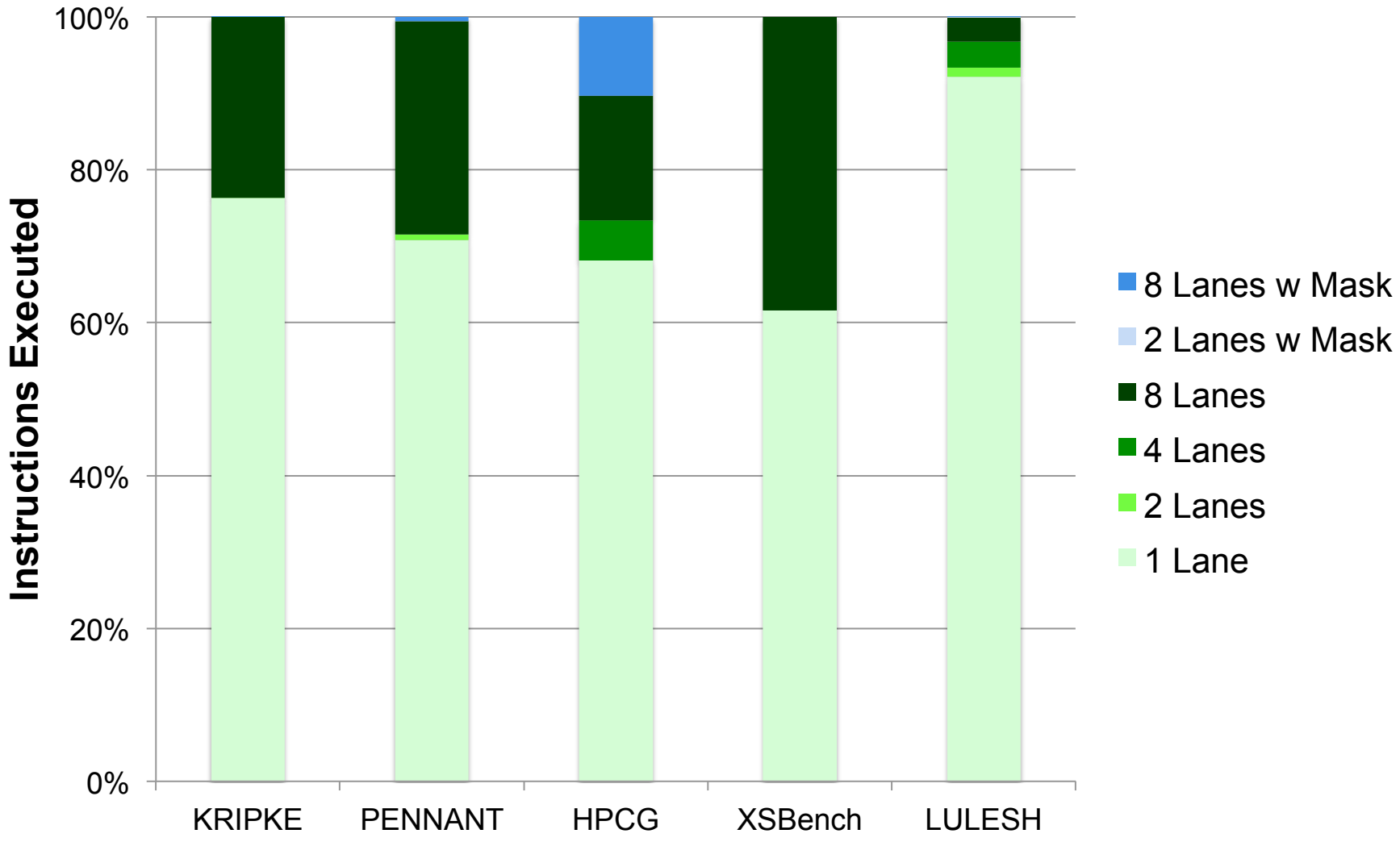


Instruction Family Breakdown



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

Vector Lane Usage inc Data Move



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

Long Term Plans

Sandia National Laboratories

- Merge the APEX analysis engines into KokkosP
- Will provide detailed multi-threaded analysis by kernel:
  - Vectorization efficiency (including vector masks)
  - Memory load/store behavior (widths of loads, stores etc)
  - Gather/Scatter analysis
  - Analysis of logical, compares etc
- Why?
  - Want to begin to make NGP-based code analysis available to vendors
  - Decide what instruction sequences etc need optimization
  - Part of Sandia's commitment to Exascale for industry

Sandia National Laboratories

DISCUSSION: PERFORMANCE TOOLS

What do you guys want us to provide for you?

Performance Tools Questions

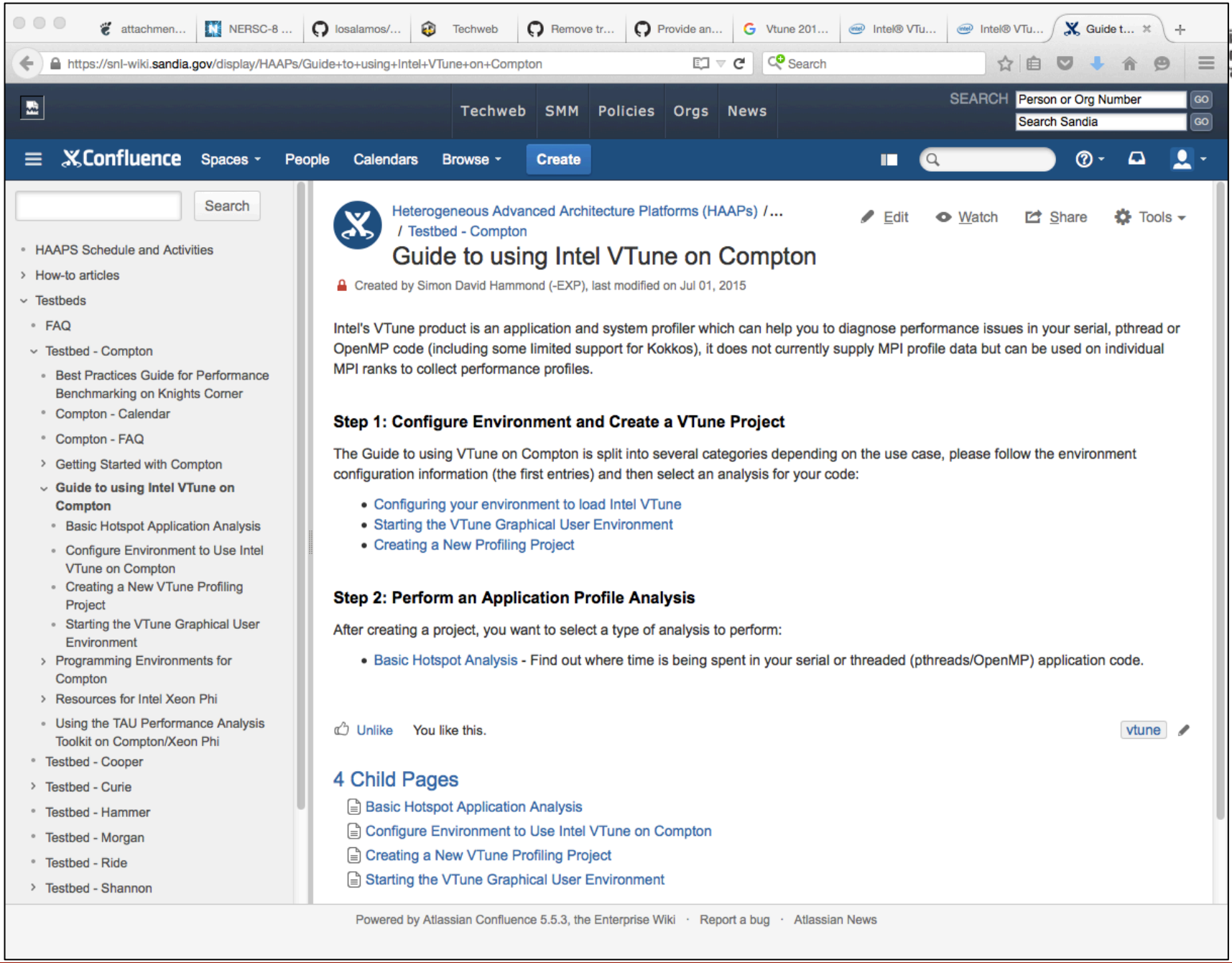
Sandia National Laboratories

- Do you have what you think you need?
- Do you use the tools mentioned or others in your daily work?
- What don't these tools do?
- Do they work with Kokkos?
- How many nodes do you need the tools to work on?

Summary & More Information?

Sandia National Laboratories

- Lots of tools available for you to use:
  - Parallel Enablement, Code Correctness and Performance Optimization
  - Most of these available on Xeon and Xeon Phi (Trinity)
- Choices for GPU and compiler
- HAAPs website: https://snl-wiki.sandia.gov/pages/viewpage.action?pageId=92176414


https://snl-wiki.sandia.gov/display/HAAPs/Guide+to+using+Intel+VTune+on+Compton
Techweb
SMM
Policies
Orgs
News
SEARCH
Person or Org Number
Search Sandia
Confluence
Spaces
People
Calendars
Browse
Create
Search
HAAPS Schedule and Activities
How-to articles
Testbeds
FAQ
Testbed - Compton
Best Practices Guide for Performance Benchmarking on Knights Corner
Compton - Calendar
Compton - FAQ
Getting Started with Compton
Guide to using Intel VTune on Compton
Basic Hotspot Application Analysis
Configure Environment to Use Intel VTune on Compton
Creating a New VTune Profiling Project
Starting the VTune Graphical User Environment
Programming Environments for Compton
Resources for Intel Xeon Phi
Using the TAU Performance Analysis Toolkit on Compton/Xeon Phi
Testbed - Cooper
Testbed - Curie
Testbed - Hammer
Testbed - Morgan
Testbed - Ride
Testbed - Shannon
Heterogeneous Advanced Architecture Platforms (HAAPs) / ...
/ Testbed - Compton
Edit
Watch
Share
Tools
Guide to using Intel VTune on Compton
Created by Simon David Hammond (-EXP), last modified on Jul 01, 2015
Intel's VTune product is an application and system profiler which can help you to diagnose performance issues in your serial, pthread or OpenMP code (including some limited support for Kokkos), it does not currently supply MPI profile data but can be used on individual MPI ranks to collect performance profiles.
Step 1: Configure Environment and Create a VTune Project
The Guide to using VTune on Compton is split into several categories depending on the use case, please follow the environment configuration information (the first entries) and then select an analysis for your code:
Configuring your environment to load Intel VTune
Starting the VTune Graphical User Environment
Creating a New Profiling Project
Step 2: Perform an Application Profile Analysis
After creating a project, you want to select a type of analysis to perform:
Basic Hotspot Analysis - Find out where time is being spent in your serial or threaded (pthreads/OpenMP) application code.
Unlike
You like this.
vtune
4 Child Pages
Basic Hotspot Application Analysis
Configure Environment to Use Intel VTune on Compton
Creating a New VTune Profiling Project
Starting the VTune Graphical User Environment
Powered by Atlassian Confluence 5.5.3, the Enterprise Wiki · Report a bug · Atlassian News

One More Thing...

Sandia National Laboratories

- ATDM Test Beds coming online
  - Initial installation of OpenPOWER ATDM systems (white and ride K80)
  - Haswell (SON hansen, SRN shiller)
  - Will have ATDM specialized file systems for fast(er) compiles
- ASC Test Beds
  - ARM64 Cavium will be delivered shortly
  - AMD Fusion APU (cooper) beginning to make some progress
- Still no dates on when KNL will arrive
- Keep an eye on the test bed announce email list!
- Thanks to the incredible test bed teams

Sandia
National
Laboratories

Exceptional service in the national interest