KokkosP: Runtime Hooks for Portable Performance Analysis

S.D. Hammond, C.R. Trott, H.C. Edwards, N. Ellingwood
Center for Computing Research,
Scalable Computer Architectures
Sandia National Laboratories, NM
sdhammo@sandia.gov
Our Research Vision

“..we should have a **single application code base** which runs across multiple platforms and architectures *and* achieves as **close to 90% of the native/best programming model/solution** as possible.”

- This is *really* hard
- But .. its what our code teams expect (and deserve)
- Requires us not just to be content to write Kokkos but also to write solutions in other models to assess how well we are doing
- We have learned **a huge** amount about Kokkos (and other models) this way
Performance Tools

- One of the problems with this approach has been that we find performance differences.

- Then .. we need code ninjas to work out where some of the time is going.

- Existing performance tools have some real weaknesses:
  - They don’t understand our abstractions
  - Some don’t have a one-to-one mapping
  - Most are single-type of device
  - Non-uniformity of tools across architectures
KokkosP Hooks

- Integrated hooks directly into the Kokkos runtime
  - Parallel kernel dispatch
  - Memory allocation using Kokkos views
  - User-defined section naming

- Dynamically loaded at runtime (no LD_PRELOAD)

- Stackable (one tool calls another, calls another)
  - Record multiple items of interest in a single run
  - Easily filter

- Working on dynamic tool feedback API
  - Control threads, data placement etc
Simple Tools

KOKKOS_PROFILE_LIBRARY=/home/sdhammo/git/kokkos-profiling-github-repo/src/tools/simple-kernel-timer/kp_kernel_timer.so ./lulesh.host -s 40
Profiling Kokkos with VTune

- Programming abstractions obscure the call stacks
- Confusing identification of Parallel Regions
  - OpenMP parallel for is in a single file: Kokkos_OpenMP_Parallel.hpp
- Very long function names which users absolutely hate
Profiling Kokkos: KokkosP+VTune

- KokkosP tools add domain marking for kernels
- VTune allows filtering, zoom in, etc. based on Domain and Frames
- Track memory allocations in timeline viewer
- Domain markings can also make CUDA kernel execution visible
Profiling Kokkos: NSight

- NSight critical for performance optimization on GPUs
  - Bandwidth analysis
  - Memory access patterns
  - Stall reasons

- **Problem:** again template based abstraction layers make awful function names, even worse than in VTune for some problems
Profiling Kokkos: KokkosP+ NSight

- KokkosP uses the CUDA8-NVTX Interface
  - Named Domains in addition to named Ranges
  - Using KokkosP NVProf-Connector to pass user-provided names through
- Shows Host Regions + GPU Regions
Some Late Breaking News...

- Agreement to work on one set of runtime hooks to rule them all (with RAJA and Kokkos)

- Means a single set of connectors/hooks will work across C++ abstraction runtimes

- Collaboration with Dave Beckingsale, Jeff Keasler and Rich Hornung (LLNL)

- Prototype(y-ish) hooks already working

- Will mean a new name (Dave is in charge of suggestions .. unfortunately)
Summary

- Just a snapshot of the tools interfaces and *connectors* we have been working on
  - Connections for parallel execution patterns
  - Memory allocations
  - User-defined named regions

- Simple tools require no vendor products (all in-house)
- But .. users can also use their favorite vendor tools (VTune, Nsight, etc)

- Research prototypes:
  - Feedback API (tools dynamically feed info back to application/Kokkos)
  - Debugging connectors
  - Vectorization and Instruction Analysis (APEX LDRD at Sandia)

- [http://www.github.com/kokkos](http://www.github.com/kokkos)
Exceptional service in the national interest

http://www.github.com/kokkos