Kokkos: Present and Future

Christian R. Trott, - Center for Computing Research
Sandia National Laboratories/NM

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA-0003525.
Transitioning To Community Project

- **Core:** 15 Developers (8 SNL)
  - More code contributions from non-SNL
    - >50% of code reviews by ORNL
    - >50% of commits from non-Sandians
- **Sandia leads API design**
- **Other labs lead backend implementations**

**Kokkos Core:**  
former: [H.C. Edwards](#), D. Labreche, G. Mackey, S. Bova, D. Sunderland,

**Kokkos Kernels:**  
[S. Rajamanickam](#), L. Berger, V. Dang, N. Ellingwood, E. Harvey, B. Kelley, K. Kim, C.R. Trott, J. Wilke, S. Acer

**Kokkos Tools:**  
[D. Poliakoff](#), S. Hammond, C.R. Trott, D. Ibanez, S. Moore, L. Cannada

**Kokkos Support:**  
[C.R. Trott](#), G. Shipman, G. Lopez, G. Womeldorff,  
former: [H.C. Edwards](#), D. Labreche, Fernanda Foertter
Kokkos Uptake

ECP Critical Dependencies

- MPI 60
- LLVM 53
- C++ 41
- OpenMP 33
- CUDA 22
- HDF5 19
- LAPACK 19
- Kokkos 18
- Fortran 17
- BLAS 16
- C 14
- ALPINE 12
- hypre 11
- Trilinos 10
- DAV-SDK 9
- ADIOS 8
- VTK-m 8
- FFT 7
- Spack 7
- OpenACC 6
- MPI-IO 6
- PnetCDF 6
- zfp 5
- SUNDIALS 5

Kokkos Slack Users

- 440 registered users
- 70 Institutions
- Every continent (-Antarctica)
- Doubles every year
Exascale Readiness

Frontier/El Capitan: HIP

- Primary development at ORNL
- Many Capabilities ready
  - Some Hierarchical parallelism is waiting for compiler bugs
- PR testing for Kokkos on AMD GPUs in place
- ArborX, Cabana, LAMMPS (partially) working

Aurora: DPC++ and OpenMP 5.0

- DPC++ blocked by compiler
  - Working with Intel on it
- OpenMP 5.0 similar state as HIP

Kokkos 3.3 (Nov 2020): OpenMP 5 and HIP expected to be largely feature complete
Updates: Training Material

- Developed The Kokkos Lectures
  - 8 lectures covering most aspects of Kokkos
  - 14 hours of recordings
  - > 500 slides
  - >20 exercises
- Hosted by ECP
  - Module 8 this Friday

- Module 1: Introduction
  - Introduction, Basic Parallelism, Build System
- Module 2: Views and Spaces
  - Execution and Memory Spaces, Data Layout
- Module 3: Data Structures and MDRangePolicy
  - Tightly Nested Loops, Subviews, ScatterView,…
- Module 4: Hierarchical Parallelism
  - Nested Parallelism, Scratch Pads, Unique Token
- Module 5: Advanced Optimizations
  - Streams, Tasking and SIMD
- Module 6: Language Interoperability
  - Fortran, Python, MPI and PGAS
- Module 7: Tools
  - Profiling, Tuning, Debugging, Static Analysis
- Module 8: Kokkos Kernels
  - Dense LA, Sparse LA, Solvers, Graph Kernels

Updates: Remote Spaces and Tooling

- Tools Support is growing
  - More Native support e.g. Tau
  - Connectors to Timemory etc.
- Nsight Systems does more useful stuff with connectors
  - Rename Kernels

- Remote Spaces beta now released
  - [https://github.com/kokkos/kokkos-remote-spaces](https://github.com/kokkos/kokkos-remote-spaces)
- Support for NVSHMEM, MPI, SHMEM
- Working on Caching, aggregation etc.
- Potentially huge productivity benefits
New Capabilities: Auto Tuning

- Part of Kokkos 3.2 (released last week)
- Tuning Interface + Tools
  - Same as other hooks: they are always there, but act as no-ops without a tool
- Multi Input – multi Output tuning
  - Inputs describe problem space
  - OutputTypes describe search space
    - Sets, Ranges, Categorical
    - Logarithmic, linear
- Tuning scopes can include multiple kernels
- Tuning of internal variables in 3.3 or 3.4

Apollo Tuner for SPMV tuning:
- Rows per team
- Team Size
- Vector Length
New Capabilities: Static Analysis

- Can we catch violations of Kokkos semantics even if code would compile/run?
  - kokkos-llvm: fork of LLVM with Kokkos aware clang-tidy variant
- Three types of violating patterns:
  - compile with some backends but not others.
  - run correct with some backends but crash on others.
  - run correct with some backends but have wrong results with others!

Example: Missing function markup

```c
void fooOOPS(int i) { printf("%i\n", i); }

int main(int argc, char **argv) {
  Kokkos::initialize();
  Kokkos::parallel_for(15, KOKKOS_LAMBDA(int i) {
    fooOOPS(i);
  });
  Kokkos::finalize();
}
```

```bash
$ clang-tidy -checks=-*,kokkos-* file.cpp
<main.cpp:7:5> warning: Function 'fooOOPS' called in a lambda was missing KOKKOS_X_FUNCTION annotation. fooOOPS(i);
^  
<main.cpp:2:1> note: Function 'fooOOPS' was declared here
void fooOOPS(int i) { printf("%i\n", i); }
```
Upcoming Capabilities: Graph Interface

- Build static graphs of kernels
  - Can use CUDAGraphs as backend
  - Allows repeated dispatch
- Helps with Latency Limited codes
  - Cuts down on launch latency
  - Can leverage streams to overlap work
  - Infers overlapping from dependencies
- Prototype release planned as part of Kokkos 3.3 (November)

```cpp
const auto graph = Kokkos::Experimental::create_graph(  
    [=](auto builder) {  
      auto root = builder.get_root();  
      auto f1 = root.then_parallel_for(  
        Kokkos::RangePolicy<>(0, 1), KOKKOS_LAMBDA(long) {...});  
      auto f2a = f1.then_parallel_for(  
        Kokkos::RangePolicy<>(0, 1), KOKKOS_LAMBDA(long) {...});  
      auto f2b = f1.then_parallel_for(  
        Kokkos::RangePolicy<>(0, 1), KOKKOS_LAMBDA(long) {...});  
      builder.when_all(f2, f3).then_parallel_for(  
        Kokkos::RangePolicy<>(0, 1), KOKKOS_LAMBDA(long) {...});  
    });

for (int i = 0; i < repeats; ++i) {  
  graph.submit();  
  graph.get_execution_space().fence();  
}
```