HPC Tools for Kokkos

Introduction to DOE Performance Portability Tutorial Day 2: Advanced Users

https://www.nersc.gov/users/training/past-training-events/2024/portability-series-kokkos-apr2024/

Vivek Kale
Christian Trott
Friday, April 26, 2023
9-9:10 AM PT

“Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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-NA0003525.”

SAND2024-5783
**Tools for Kokkos Programming**

**Kokkos Tools** = A set of event-callback tool libraries, and the common infrastructure to support them

### Manual
1. **Profile:**
   - TPL Vendor connectors, e.g., nvtx, roctx
   - Timer
   - Space-time-stack
   - Memory usage
2. **Debug:**
   - kernel logging
   - memory events

### Automated
- **Auto-analyze:**
  - Test case generation
  - Performance insights
  - HPC System Monitoring
- **Auto-fix:**
  - fault-correction
  - auto-tuning

→ Almost all tools use the Kokkos Tool interface hooking into Kokkos core (Kokkos_Profiling.hpp)

https://github.com/kokkos/kokkos-tools → □Fork me!

nvtx-connector: HPCToolkit (John Mellor-Crummey et al)

tau/apex (Kevin Huck, Sameer Shende et al)
#include <Kokkos_Core.hpp>

int main(int argc, char* argv[]) {
    Kokkos::initialize(argc, argv);
    Kokkos::View<double*> u(102), unew(102);
    Kokkos::View<double*>::HostMirror host_u = create_mirror_view(u);
    deep_copy(host_u, 1.0);
    int tstep = 0;
    while (tstep < 1000) {
        Kokkos::Profiling::pushRegion("myCoolTimestepRegion");
        deep_copy(u, host_u);
        parallel_for(102, LAMBDA(int i) {
            unew[i] = (u[i-1] + u[i] + u[i+1])/3.0;
        });
        deep_copy(host_u, unew);
        Kokkos::Profiling::popRegion("myCoolTimestepRegion");
        tstep++;
    }
    Kokkos::finalize();
}
Kokkos Tools: Available Set of Tools

1. Developing a tool of Kokkos Tools requires implementing a small number of event callbacks corresponding to Kokkos user functions, e.g., parallel_for, Kokkos_initialize

2. Each tool operates independently, though any subset of Kokkos Tools can be built as one library.

3. The Kokkos Tools implementation facilitates low instrumentation overhead through capabilities such as its sampling and filtering utilities.

Agenda for the Session

1. 00-9:10 AM PT: Overview of tools for Kokkos including basic tools - Vivek Kale
2. 9:10-10AM PT: HPCToolkit for Kokkos applications - John Mellor-Crummey
3. 10:00 -10:30 AM PT: autotuning with Apex connector - Kevin Huck
4. 10:30 AM - 11:00 AM PT: all hands-on with
   ○ nvtx-connector
   ○ HPCToolkit
   ○ tau_exec and apex_exec - Kevin and Sameer
5. 11-12 PM PT: Build your own Kokkos tool - Daniel Arndt
# Aiding Kokkos Parallel Programming via Kokkos Tools Set

## Sample Output

![Sample Output Image]

## Quick Start

### Setup

**From source:**

**Install**

```bash
git clone github.com/kokkos/kokkos-tools;
```

**Build**

```bash
→ Make
cd toolName; make;
→ CMake
mkdir build; cd build; cmake ..; make; make install;
```

### Use

```bash
export KOKKOS_TOOLS_LIBS="toolName.so; toolName2.so;";
./myKokkosApp.exe
```

## Contact

Vivek Kale ([vlkale@sandia.gov](mailto:vlkale@sandia.gov)) and Christian Trott ([crtrott@sandia.gov](mailto:crtrott@sandia.gov))