Sophisticated Tool Support to Aid Performance Portable Programming

David Boehme, Kevin Huck, Shravan Kale, Vivek Kale, Vanessa Surjadidjaja

Sandia National Laboratories, Lawrence Livermore National Laboratory, University of Oregon

Tool Support for Performance Portable Programs

- Background: HPC Software for performance portable programming using abstractions for parallelism, e.g., Kokkos, RAJA, are emerging vehicles for Scientific Software run on supercomputers having node-level heterogeneity.
- Challenge: For this HPC Software's sustainability, it must aid programmer productivity in developing performant applications.
- One solution: Use basic tools support for profiling and debugging that is associated with performance portable library.
- A limitation: Large programmer effort still needed to manually assess and tune their applications given, e.g., timings and logs.
- Opportunity to improve: Performance portable programs benefit from more sophisticated activities, particularly: performance analysis and auto-tuning of applications run on supercomputers, as well as performance monitoring of a collection of applications run on a supercomputer, i.e., HPC Systems.
- Approach: Provide easy-to-use and low-overhead tool support offering capabilities for sophisticated activities.
- Through a focus on Kokkos, this poster aims to showcase a part of Kokkos Tools that offer capabilities for these sophisticated activities, given the alternative of tools specific to lower level programming libraries for parallelism.

Kokkos Tools Common Infrastructure

1. Kokkos Tools: a component of the Kokkos ecosystem supporting productivity for Kokkos programming through built tool libraries containing Kokkos function event callbacks, i.e., tool connectors, along with groundwork to support it.
2. Using it for Kokkos involves simply setting `KOKKOS_TOOLS_LIBS` to an appropriate sequence of utilities and connectors.
3. Each connector operates independently and is self-contained; a subset of Kokkos Tools can be built as a single library.
4. A part of Kokkos Tools helps said sophisticated activities. It comprises of:
   (a) a set of community and in-house tool connectors for said sophisticated activities; and
   (b) for use by tool connectors of (a), common utilities, e.g., kernel filter, sampler, and a common tool support infrastructure
5. Kokkos Tools common utilities offers common set of capabilities for easily developing any third-party tool connectors and the infrastructure allows connectors to perform arbitrarily complex actions upon the event of Kokkos kernel or function invocation.
6. The three Kokkos Tools connectors for sophisticated activities are (1) Caliper for performance analysis of applications, (2) Apex for auto-tuning applications, and (3) LDMS for performance monitoring of HPC Systems.

`https://github.com/kokkos/kokkos-tools`

Performance Analysis with Caliper

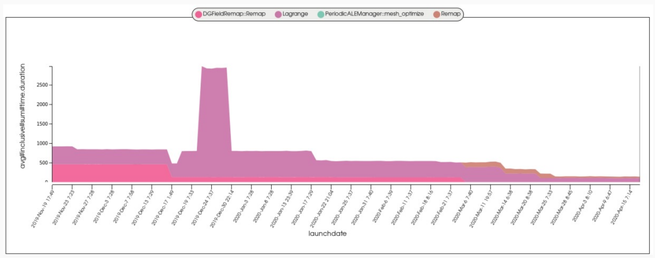


Figure 1: Nightly performance regression testing of a large physics code with Caliper.

Caliper is an instrumentation and performance profiling library. It measures user-defined regions in C++ HPC applications through its source-code annotation API or the Kokkos profiling interface.

Ensemble Analysis

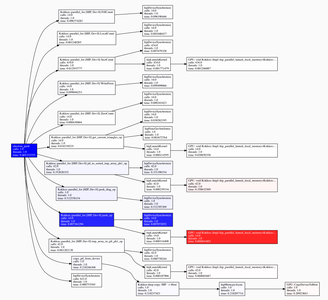
- Caliper is particularly well-suited for ensemble analysis, such as scaling studies or comparing different program configurations.
- To support ensemble analyses, Caliper records program metadata like build and execution configurations as well as application input options.

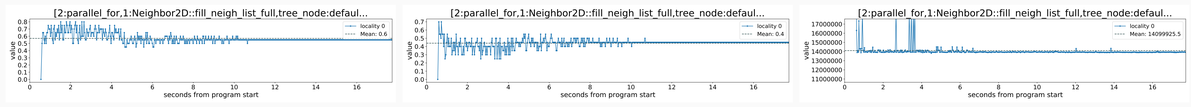
`https://github.com/LLNL/Caliper`

Auto-tuning with APEX

The Autonomic Performance Environment for eXascale (APEX) is a performance tool designed for asynchronous runtimes such as HPX, Pthreads, OpenMP, and GPU-accelerated codes. As such, it is well suited for measuring both Kokkos and the back-ends available in Kokkos. APEX is integrated with both the Kokkos Tools interface as well as the Kokkos tuning interface to both measure and runtime tune Kokkos applications.

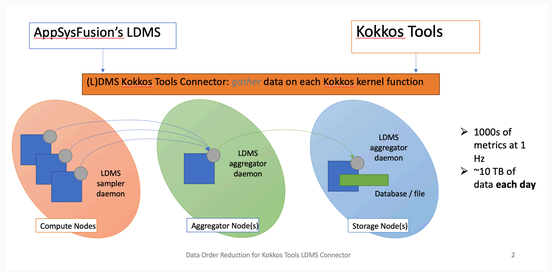
- APEX measures Kokkos data allocations, frees, and deep copies.
- APEX measures `parallel_for`, `parallel_reduce`, `parallel_scan` constructs.
- APEX measures the back-end exectution, and connects task dependency across physical resources (threads, GPUs) – see figure, right.
- APEX performs runtime autotuning on standard kernels to determine optimal vector length and team size for `TeamPolicy` – see figure, below, for example with ExaMiniMD.





`https://github.com/UO-OACISS/apex/`

Performance Monitoring with LDMS



- The Lightweight Data Monitoring System, or LDMS, is a performance monitoring tool for an HPC System.
- Its LDMS Kokkos Tools connector enables direct analysis of Kokkos applications in terms of granularity and specificity of Kokkos kernel functions, e.g., analysis of memory usage of a `Kokkos::parallel_for()` in the scope of an HPC System.
- Figure illustrates the conceptual workflow of LDMS connector, and its use of the common Kokkos Tools infrastructure and sampler utility.
- Sampler offers data order reduction and time efficiency for LDMS data collection, and it is showing performance benefit to DoE applications using the LDMS connector.

`https://github.com/ovis-hpc/ovis`

Conclusions and Next Steps

Summary

- Showcased infrastructure of Kokkos Tools and 3 sophisticated Kokkos Tools connectors: (1) Caliper; (2) Apex; and (3) LDMS.
- These open-source connectors offer potential for performance analysis and auto-tuning to quickly obtain high performance for Kokkos programs on exascale systems.

Future Work

- Consider AI/ML engines for auto-analysis and tuning
- Test with MPI+Kokkos applications
- Reduce overheads on Apex and Caliper by removing overly conservative fencing and by trying sampler.
- LDMS connector with feedback

Contact: boehme3@llnl.gov, khuck@cs.uoregon.edu, shravank@cs.uoregon.edu, vlkale@sandia.gov, vsurjad@sandia.gov