GSoC 2024 Proposal

Implement Differentiating of the Kokkos Framework in Clad

Mentors: Vaibhav Thakkar, Vassil Vassilev, Petro Zarytskyi

Contact

Name: Atell (Yehor) Krasnopolskyi
Email: delta_atell@protonmail.com
Github: gojakuch
Location: Germany (eligible to work)
Citizenship: Ukraine
Timezone: CET, UTC+1 (Berlin)

Candidate's Prior Experience

IRIS-HEP Fellowships 2022 & 2023:

- Jet Reconstruction with Julia, mentors: Benedikt Hegner (CERN), Graeme A Stewart (CERN)
- Julia for Analysis Grand Challenge, mentors: Jerry Ling (Harvard University), Alex Held (UWM)

Contributions to the project so far:

- Add Kokkos unit tests #826
- [cmake] Provide the DISABLE_LLVM_LINK_LLVM_DYLIB flag for unittests #819 (merged)

Project Overview

In mathematics and computer algebra, automatic differentiation (AD) is a set of techniques to numerically evaluate the derivative of a function specified by a computer program. Automatic differentiation is an alternative technique to Symbolic differentiation and Numerical differentiation (the method of finite differences). Clad is based on Clang which provides the necessary facilities for code transformation. The AD library can differentiate non-trivial functions, to find a partial derivative for trivial cases and has good unit test coverage.

The Kokkos C++ Performance Portability Ecosystem is a production level solution for writing modern C++ applications in a hardware-agnostic way. It is part of the US Department of Energies Exascale Project – the leading effort in the US to prepare the HPC community for the next generation of supercomputing platforms. The Ecosystem consists of multiple libraries addressing the primary concerns for developing and maintaining applications in a portable way. The three main components are the Kokkos Core Programming Model, the Kokkos Kernels Math Libraries and the Kokkos Profiling and Debugging Tools.

The Kokkos framework is used in several domains including climate modelling where gradients are an important part of the simulation process. This project aims at teaching Clad to differentiate Kokkos entities in a performance-portable way.

Implementation Details

The goal is to implement the differentiation of the Kokkos high-performance computing framework including the support of:

- Kokkos functors,
- Kokkos lambdas,
- Kokkos methods such as parallel_for, parallel_reduce and deep_copy,
- as well as the general support for Kokkos::View data structures,
- Enhance existing benchmarks demonstrating effectiveness of Clad for Kokkos

The additional aim of the project is to implement a generic approach to support any C++ library (starting with Kokkos) in such a way that the core of Clad is invariant to the internals of the library, but any Clad user can add it in a pluggable format for individual use cases. This ensures Clad's usability for bigger projects that may include a lot of libraries.

The set-off points for the project should be the existing "Kokkos-aware Clad" PR #783 and the test cases from my PR #826 for Kokkos unit tests give a basis of what should definitely be implemented and serve as an initial list of use cases for Kokkos-aware Clad. The first steps would include implementing the features of PR #783 one by one by designing the more generic approach mentioned earlier while using PR #826 as a reference.

PR #826 mentioned above illustrates the possible usage of Clad with Kokkos lambda functions, functions accessing Kokkos views and copying them, as well as functions calling parallel loops (parallel_for, parallel_reduce).

Further steps include developing the benchmarks, optimisation, polishing and perfecting the desired general approach of making a library differentiable for Clad. This way, the project acquires a wider set of applications, while not losing Kokkos as the initial target.
Thus, a possible plan may look like this:

- Some time will be needed for diving deeper into the insides of Clad and understanding more about the relevant Kokkos features
- Creating a general approach to making a library differentiable for Clad
- Implementing the features of PR #783 one by one with the desired approach in mind
- Checking against the existing unit tests provided in PR #826
- Developing the benchmarks, more unit tests that address the body of the generated derivatives (there are only value tests currently), optimisation
- Polishing and perfecting the desired general approach

To get into technical details, Clad uses the propagator system in its workings, that is the pushforward method and the pullback method. In both forward and reverse modes, the methods are called to differentiate other functions called inside the requested function (although the forward mode only uses the pushforward method, as far as I am aware). The idea behind the pushforward method is to associate the partial derivative, $\partial v/\partial x$, w.r.t. the requested parameter, x, with every expression, v, starting with binding 1 as the derivative of the argument w.r.t. itself. The idea of the pullback is similar, yet here the starting point is the adjoint value of the function. Pullback pairs each variable u with $\partial f/\partial u$. The pullback method, for instance, generalises the gradient method itself, in the sense that it accepts the base adjoint value as an additional parameter (while the gradient sets it to 1 by default). To make Kokkos differentiable in Clad, one would need to be able to propagate pullbacks and pushforwards through its constructs, which is one of the goals of this project.

Timeline

Community Bonding Period	
01.05.2024-26.05.2024	Engage with the community. Pick and solve some issues from GitHub to get a deeper understanding of Clad's internals. Reach out to experts and mentors for opinions on the ideal project structure. Establish regular meetings with the mentors. Set up the development environment. Set up external tools as debuggers, debug versions of the compiler. Enhance the unit tests provided in PR #826 to cover more deliverables of the project and outline the expected work better.
Coding period begins	
Week 1 27.05.2024-02.06.2024	First attempts to make clad work with Kokkos views, clarified project structure, first draft of the generic approach mentioned above (possibly using custom derivatives). **Deliverable:** initial support of Kokkos views, a general approach of making a library differentiable for Clad.

Week 2 03.06.2024-09.06.2024	Support for Kokkos views and primitive operations. **Deliverable:** advance the support of Kokkos views with assignment by index, element access (both in forward and reverse mode).
Week 3 10.06.2024-16.06.2024	Support for copying Kokkos views. **Deliverable:** differentiation of the code with the deep_copy method (forward mode only).
Week 4 17.06.2024-23.06.2024	Support for basic Kokkos parallel operations. **Deliverable:** differentiation of Kokkos lambdas and parallel_for (forward mode only).
Week 5 24.06.2024-30.06.2024	Support for Kokkos functors. **Deliverable:** full support for Kokkos functors and Kokkos inline functions (forward mode only).
Week 6 01.07.2024-07.07.2024	Finish reimplementing the features of PR #783. **Deliverable:** support for Kokkos subviews and everything else PR #783 provides (all compatible layouts, memory spaces and host spaces but using the new approach) in the forward mode only.
Week 7 08.07.2024-14.07.2024	Buffer week in case of delays in reviews or implementation challenges.
Midterm Evaluations	
Week 8 15.07.2024-21.07.2024	Support for Kokkos parallel_reduce. **Deliverable:** differentiation of Kokkos parallel_reduce with policies of user's choice (forward mode only).
Week 9-10 22.07.2024-28.07.2024 29.07.2024-04.08.2024	Reverse mode. **Deliverable:** every feature supported by the forward mode should be supported by the reverse mode as well.
Week 11 05.08.2024-11.08.2024	Update tests and benchmarking, make unit tests keep up with the progress. **Deliverable:** add tests that check the generated code as opposed to value-only tests, develop benchmarks demonstrating the effectiveness of the current approach of Clad for Kokkos.
Week 12 12.08.2024-18.08.2024	Develop enhanced test case examples for extended features that will eventually be used to work on making them differentiable with Clad. **Deliverable:** test cases with functions that take Kokkos views as parameters, return them, or both (possibly with clad::jacobian); test cases with other View-like types such as Kokkos::DynamicView, Kokkos::OffsetView, and Kokkos::DynRankView.
Week 13 19.08.2024-25.08.2024	Developing documentation, presenting the work. **Deliverable:** demonstrated reduction of the binary sizes, a blog post about the achieved results; presentation at the compiler-research.org team meeting.

Candidate's Other Commitments

The only other obligatory commitment of any importance during the coding period is my studies at the University of Wuerzburg (June only). This means having 2-3 subjects with

supposedly non-differentiated grading tests at the end (early July). Attending lectures and practice sessions is not obligatory and from my experience of the previous 3 semesters, regular attendance is not necessary. Thus, I will occasionally show up to the lectures when I have time and I will need to write the final tests. However, taking my experience into account, I am sure that this commitment is not going to prevent me from achieving the project goals. Moreover, the courses I am taking this semester were selected with GSoC in mind and, therefore, should not be too time-consuming.