Google Summer of Code

GSoC 2023 Project Report

Implement vector mode in forward mode automatic differentiation in Clad

- **Student**: Vaibhav Thakkar
- **Github**: @vaithak
- **Organisation:** CERN-HSF
- **Mentors**: Vassil Vassilev, Parth Arora
- **Project repository**: Clad
- **Project proposal**: Proposal link

Overview of the Project

Aim of the Project

Clad is an open source plugin to the Clang compiler that detects calls to differentiate a defined function from the parsed Abstract Syntax Tree(AST), generates code that is the derivative the function and modifies the AST to insert the generated code.
Clad works using the concept of Automatic Differentiation(AD), which in mathematics and computer algebra is a set of techniques to numerically evaluate the derivative of a function specified by a computer program.
Vector mode support will facilitate the computation of gradients using the forward mode AD in a single pass and thus without explicitly performing differentiation n times for n function arguments. The major benefit of using vector mode is that computationally expensive operations do not need to be recomputed n times for n function arguments.

Project Deliverables

This project proposes to add the following features to Clad:

1. Extend and generalize the ForwardModeVisitor to produce a single function with the directional derivatives.
2. Add a new mode to the top-level clad interface clad::differentiate for vector mode.
3. Document the above features and write unit tests for them.

Contributions

The major PRs I contributed in the GSoC period are mentioned here. All of them have been merged.

PR	Description
#565	Add initial basic support for vectorized forward mode AD
#572	Split ForwardMode into separate classes
#576	Ensure git-clang-format is run on PR
#577	Enhance vector fwd mode to differentiate w.r.t selected params
#579	Run clang tidy check on PR
#583	Improve AD function interfaces with bitmasked options
#607	Fix LLVM assertion errors for vector mode
#609	Add matrix class in clad
#614	Add support for array arguments in vector mode

To see all the PRs contributed to Clad by me, checkout this link

Results

Here is an example that demonstrates how one can request clad to use vector mode to differentiate a function in forward mode.

```
#include "clad/Differentiator/Differentiator.h"

double prod(double x, double y) { return x * y; }

int main(){
    auto df = clad::differentiate<clad::opts::vector_mode>(prod);
    double x = 3, y = 4;
    double dx = 0, dy = 0;
    df.execute(x, y, &dx, &dy);
    printf("d_x = %.2f, d_y = %.2f\n", dx, dy);  // Results in: d_x = 4.00, d_y = 3.00
}
```

Thus, the calling convention is to use `clad::differentiate<clad::opts::vector_mode>(...)` instead of the usual calling convention, `clad::differentiate(...)`.

To read more about vector forward mode in clad, please visit here

Link to my blogpost explaining vectorized forward mode AD

Future Work

Support for Vector Forward Mode AD is still not complete, some essential features to be added:

- Differentiation of functions containing call expressions - this can be either std functions like `pow` or `exp`, or calling user-defined functions.
- Support for object oriented features in vector mode; this is to essentially allow differentiating method calls or functors.

- Improving efficiency to match that of vectorized AD implementations in other libraries, for ex: enzyme.

Some experiements on auto-vectorization

As suggested in LLVM's documentation on auto-vectorization, ran some diagnostic tests with `-Rpass=loop-vectorize` flag to ensure that `clad::array` and `clad::matrix` operations are vectorized by the compiler.
The code programs used for this were the same as those present in `clad/demos` and `clad/tests` for vector forward mode.