Compiler Research

Summary of Activities 2024

Vassil Vassilev

12.12.2024

Clad — Enabling Differentiable Programming in Science

Source Transformation AD With Clad

- Development
  - Enable CUDA
  - Extend Kokkos Support
  - Rework Jacobians
  - Implement Varied Analysis
  - Support operators
  - Enhance support of std::array, std::vector, std::tie,
  - Constexpr support and clad::immediate_mode
- Scientific use-cases
  - Towards supporting STL and Thrust
- Next milestone v1.8 is planned in the end of the month

Source Transformation AD With Clad

- Towards enabling clad in the field of High-Energy Physics
  - Differentiable RooFit: Worked on enabling several large workflows (order of 100K lines of code)
  - Differentiable Combine: Adoption of the technique in CMS Combine
  - Promising speedups

C++ as a service — rapid software development and dynamic interoperability with Python and beyond

Hands on details can be seen in our showcase presentation.

Status. Cling

- Being upgraded to llvm18 — complete. Released v1.2
- Upstreamed [Serialization] Support loading template specializations lazily

Status. Clang-Repl

- A good chunk of autoloading facilities is open against llvm in PR109913
- Making slow progress on:
  - PR84769 — [clang-repl] Implement Value pretty printing for containers. Value Handling (RFC)
  - Simplified the value printing logic, broke cuda, working on fixing it

The goal is to provide better stability and robustness which can later cling can reuse.

Status. CppInterOp

- Enabled Wasm
- Enabled llvm-19
- Merged Add a libclang-style C API (337)
- Improved documentation
- Added support of externally created interpreters
- Started gradual adoption in ROOT
- Upstreaming PR 308 in llvm [ORC] Add Auto-Loading DyLib Feature with Symbol Resolution
  - This is the last missing element to deprecate completely xeus-cling in favor of xeus-cpp

Status. Xeus-Cpp

- Compled on adding LLM support
- Working on merging more infrastructure xeus-clang-repl into xeus-cpp

Status. Xeus-Clang-Repl

- No updates

GSoC, IRIS-HEP, HSF-India 2024 Summary

CppInterOp in ROOT

Aaron Jomy

Research Intern at CERN

Info

- Jan 2024 -
- Adoption of CppInterOp in ROOT
- Rebased cppyy on our forks
- Implemented CI support for our cppyy forks
- Added template support to CppInterOp
- Brought the migration from 188 passing tests to 276 passing tests (out of 504)

Adopting CppInterOp in cppyy

Vipul Cariappa

Ramaiah University, India,

HSF-India

Info

- September 2024 -
- Main focus is moving forward with replacing of ROOT in cppyy.
- Started with ~276 passing tests, now 328 passing out of 504
- Improved various of facilities in CppInterOp in the areas such of templates instantiations and global operators.
- Improved the interactive dynamic differential debugging capabilities to enable debugging complex workflows such as cppyy

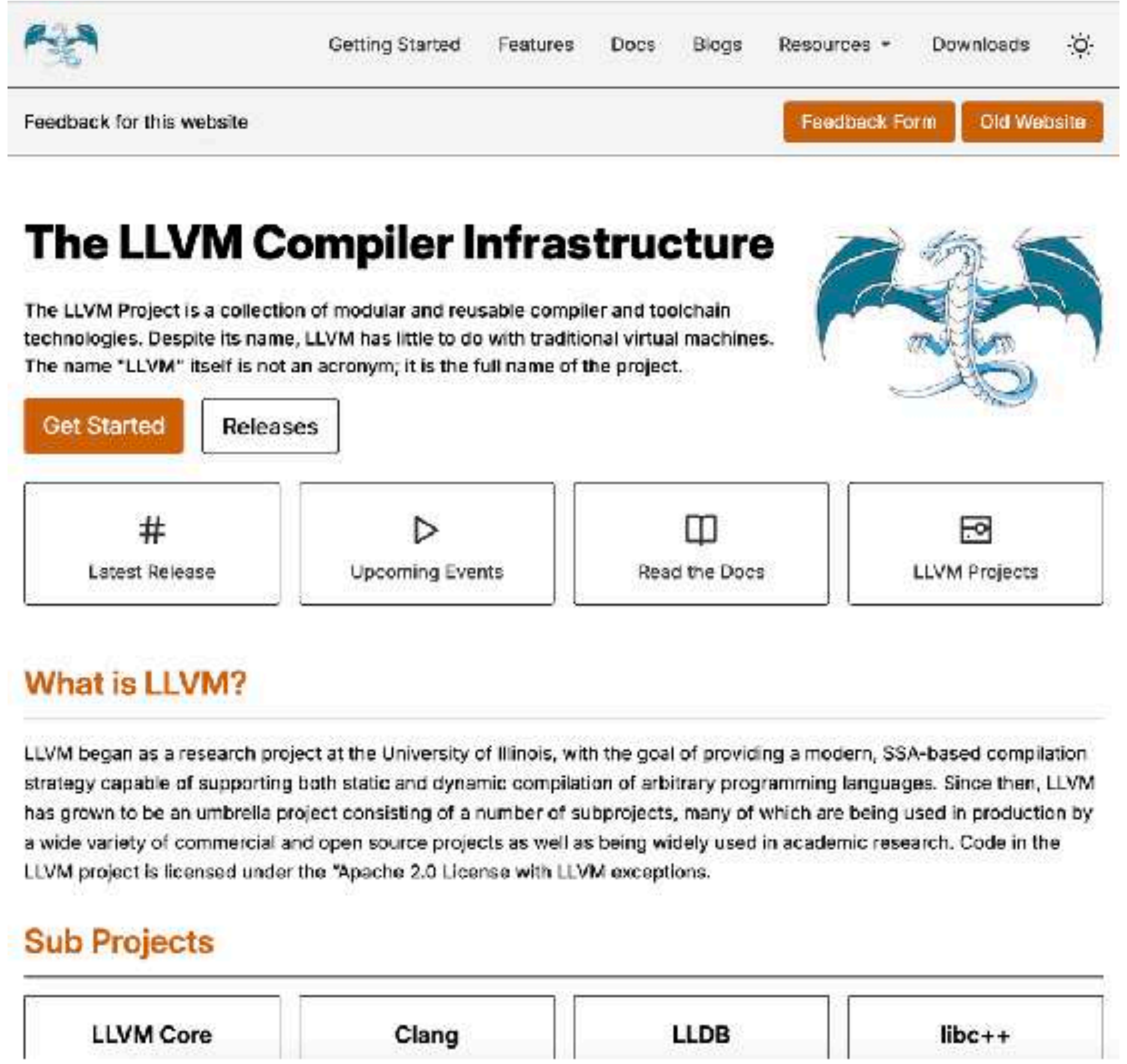
LLVM.org Website Redesign

- May-Nov 2024 (ongoing)
- Reworked the old website
- Used Hugo — a static website generator
- Developed a new reusable, modern and mobile-friendly theme
- Organized a community process in gathering and addressing feedback
- To be enabled by default
- Blog entry

Chaitanya Shahare

GSoC24 National Institute of Technology Srinagar, India

Info



Feedback for this website

The LLVM Compiler Infrastructure

The LLVM Project is a collection of modular and reusable compiler and toolchain technologies. Despite its name, LLVM has little to do with traditional virtual machines. The name "LLVM" itself is not an acronym; it is the full name of the project.

What is LLVM?

LLVM began as a research project at the University of Illinois, with the goal of providing a modern, SSA-based compilation strategy capable of supporting both static and dynamic compilation of arbitrary programming languages. Since then, LLVM has grown to be an umbrella project consisting of a number of subprojects, many of which are being used in production by a wide variety of commercial and open source projects as well as being widely used in academic research. Code in the LLVM project is licensed under the "Apache 2.0 License with LLVM exceptions.

Sub Projects

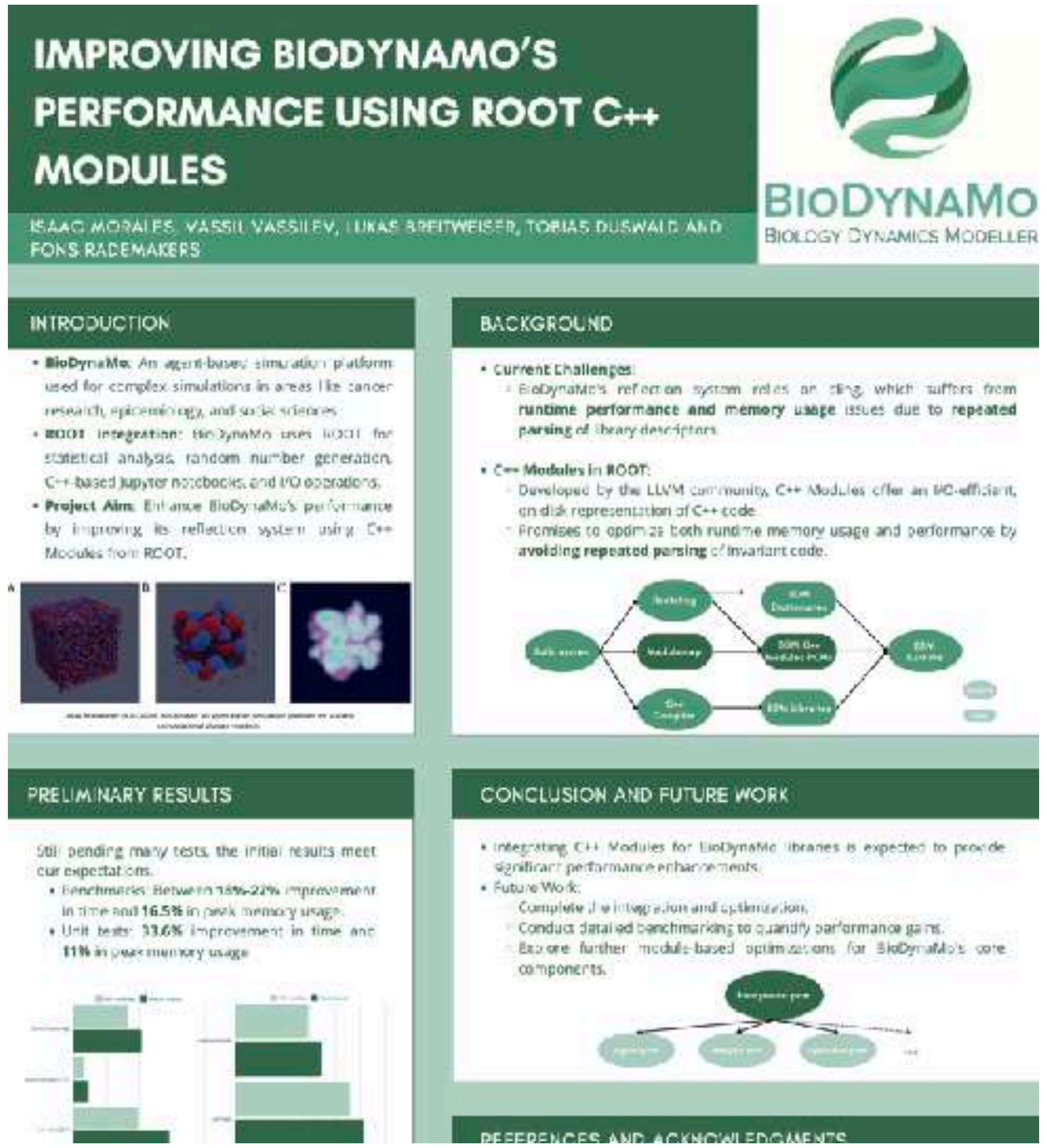
Improving performance of BioDynaMo using ROOT C++ Modules.

Isaac M. Santana

GSoC24, University of Granada, ES

Info

- May-Nov 2024
- Enabled C++ Modules as optimization data structure to help with slow startup times
- Presented the work at the 4th Mode Workshop in Valencia
- Most of the work is merged in BioDynaMo upstream a few elements still under review
- Blog entry



IMPROVING BIODYNAMO'S PERFORMANCE USING ROOT C++ MODULES

ISAAC MORALES, VASSIL VASSILEV, LUKAS BREITWEISER, TOBIAS DUSWALD AND FONS RADEMAKERS

BIODYNAMO
BIOLOGY DYNAMICS MODELLER

INTRODUCTION

- **BioDynaMo:** An agent-based simulation platform used for complex simulations in areas like cancer research, epidemiology, and social sciences.
- **ROOT Integration:** BioDynaMo uses ROOT for statistical analysis, random number generation, C++-based Jupyter notebooks, and I/O operations.
- **Project Aim:** Enhance BioDynaMo's performance by improving its reflection system using C++ Modules from ROOT.

BACKGROUND

- **Current Challenges:**
  - BioDynaMo's reflection system relies on cling, which suffers from **runtime performance and memory usage** issues due to **repeated parsing** of library descriptors.
- **C++ Modules in ROOT:**
  - Developed by the LLVM community, C++ Modules offer an I/O-efficient, on-disk representation of C++ code.
  - Promises to optimize both runtime memory usage and performance by **avoiding repeated parsing** of invariant code.

PRELIMINARY RESULTS

Still pending many tests, the initial results meet our expectations.

- Benchmarks: Between 15%-22% improvement in time and **16.5%** in peak memory usage.
- Unit tests: **33.6%** improvement in time and **11%** in peak memory usage.

CONCLUSION AND FUTURE WORK

- Integrating C++ Modules for BioDynaMo libraries is expected to provide significant performance enhancements.
- Future Work:
  - Complete the integration and optimization.
  - Conduct detailed benchmarking to quantify performance gains.
  - Explore further module-based optimizations for BioDynaMo's core components.

REFERENCES AND ACKNOWLEDGMENTS

ROOT superbuilds

Pavlo Svirin

GSoC24, Kyiv University, UA

Info

- May 2024 -
- Provide a way to build ROOT piecewise.
- Reworked the cmake infrastructure to allow for building each component in isolation
- Improved C++ module definitions into separate modulemap files
- Work yet to be merged in ROOT
- Blog entry

Integrate a Large Language Model with the xeus-cpp Jupyter kernel

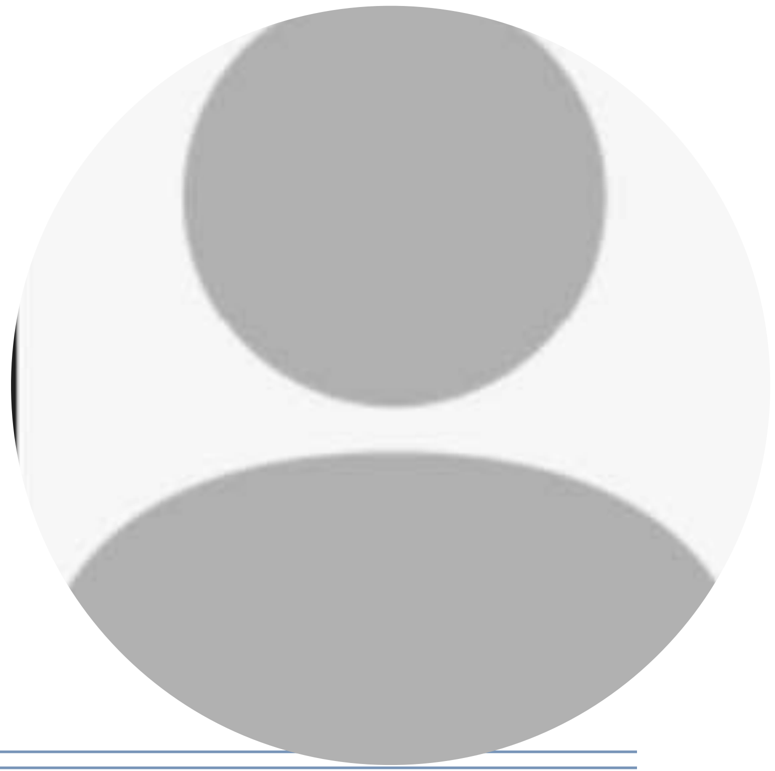
Tharun Anandh

GSoC24, National Institute of Technology, Tiruchirapalli, India

Info

- May-Nov 2024
- Xeus-cpp is a C++ execution engine for Jupyter. The goal of the project was to integrate a LLM service allowing people to interact with when developing code
- Implemented a general approach to integrate large set of LLMs
- Blog entry

Support clang plugins on Windows



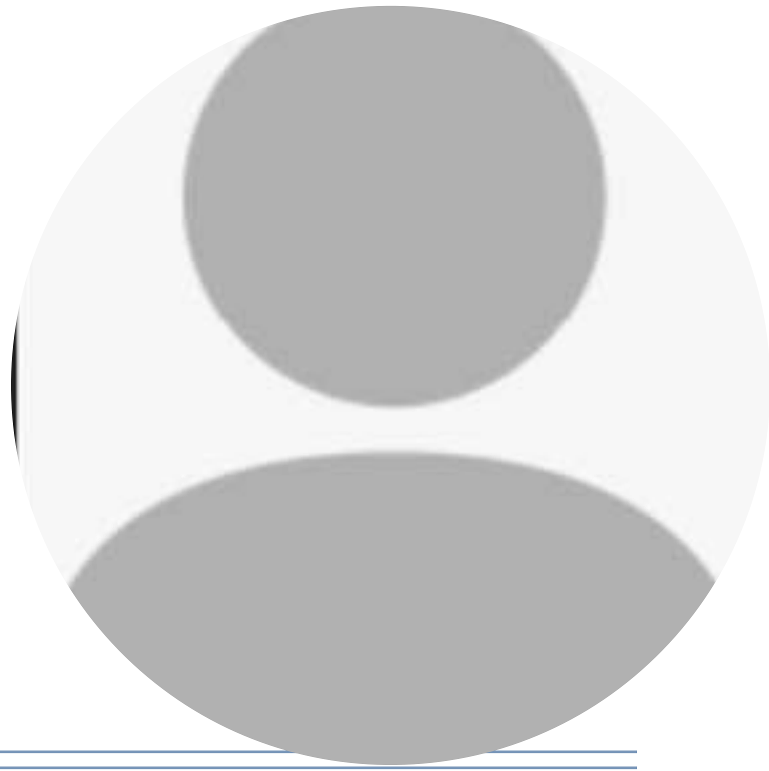
Thomas Fransham

GSoC24, UK

Info

- May-ongoing 2024
- Clang plugins (Clad included) do not work on windows because LLVM interfaces need to be annotated as “public”
- This is a huge project requiring touching thousands of header files. Some changes are trivial some not.
- Large portion of work has been done (~1/3). Working on CI.
- Demonstrated decrease of on disk memory
- Meta issue

Out-Of-Process execution for Clang-Repl



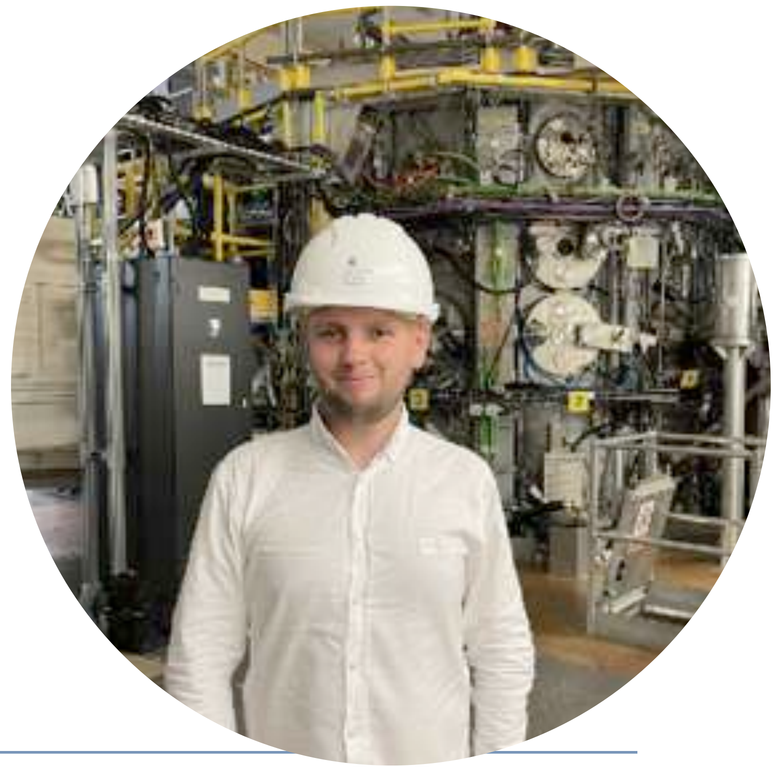
Sahil Patidar

GSoC24, Vindhya Institute of Technology, India

Info

- May-ongoing 2024
- ClangRepl run the user code as part of the current process. Out-of-process execution splits the user code from the process executing the binary improving the crash resilience and security
- clang-repl --oop-executor=path/to/llvm-jitlink-executor --orc-runtime=path/to/liborc_rt.a
- [ORC] Add Auto-Loading DyLib Feature with Symbol Resolution
- Enable Auto-Loading Support in Root/LLVM
- Blog entry

Continuous Integration, CppInterOp, Xeus-Cpp



Matthew Barton

Open Source Contributor

Info

- Development of the continuous integration system for our ecosystem including clad, cppyy, CPyCpyy, cling-backend, CppInterOp and LLVM
- Increased testing coverage for CppInterOp
- Improved the Wasm Infrastructure
- Added llvm 18 and 19 support to CppInterOp
- Improved Windows support for CppInterOp
- Fixed all warnings so we could treat all future warnings as errors in CppInterOp and xeus-cpp
- Added llvm 18 support to Clad for Linux
- Got CppInterOp available for multiple platforms for conda and in emscripren forge

Xeus-Cpp, Wasm, Xeus

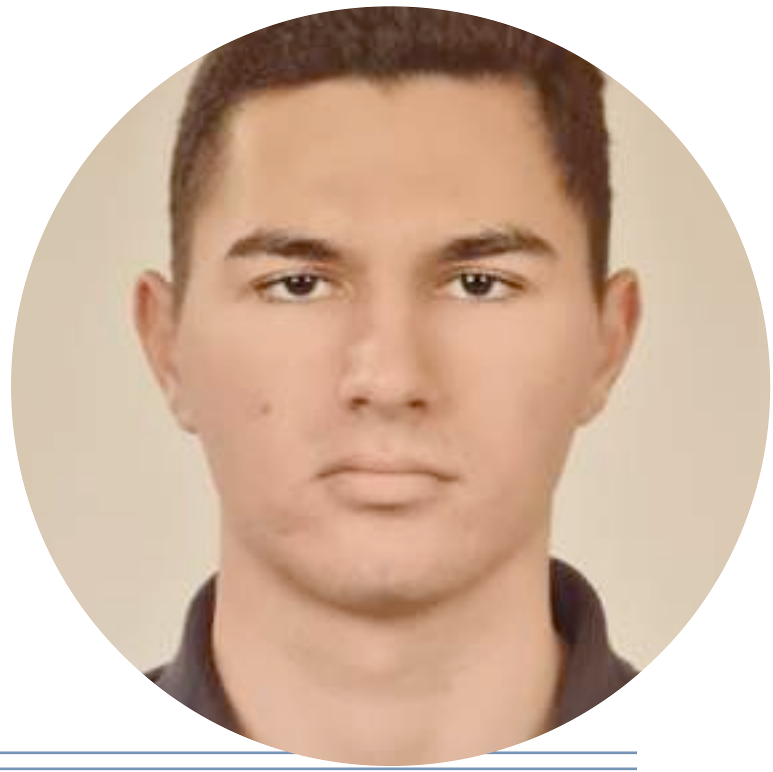
Anutosh Bhat

Open Source Contributor to xeus-cpp, CppInterOp, India

Info

- Jan-ongoing 2024
- Maintaining work on xeus-cpp, Xeus and Xeus-zmq
- Enabled clang-repl in Wasm
- Improved CppInterOp for emscripten
- Packaging

Add support for consteval and constexpr functions in Clad



Mihail Mihov

GSoC24, Stara Zagora Math High School, BG

Info

- May-Nov 2024
- C++ extensively uses compile-time metaprogramming with constexpr and consteval keywords which force the compiler frontend to run functions.
- Enabled constexpr and consteval support in Clad including making CladFunction constexpr-friendly
- Implemented clad::immediate_mode
- Blog entry

Implement Differentiating of the Kokkos Framework in Clad

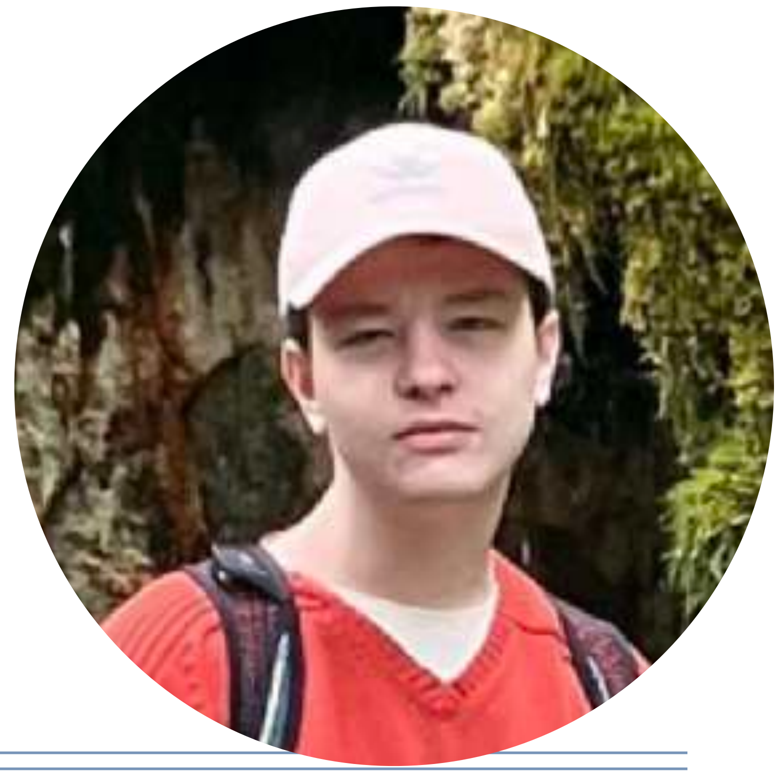
Atell Yehor Krasnopolski

GSoC24, University of Wuerzburg, DE

Info

- May-Nov 2024
- Kokkos is a C++ library that enables writing performance portable codes.
- Developed an extensible system for defining library-specific push forward and pullback operators in Clad
- Added support for several STL entities such as std::array and std::vector
- Lambda support still to be completed
- Presented the work at the 4th Mode Workshop in Valencia
- Blog entry

Optimizing automatic differentiation using activity analysis



Maksym Andriichuk

GSoC24, University of Wuerzburg, DE

Info

- May-ongoing 2024
- Presented the work at the 4th Mode Workshop in Valencia
- Implemented useful analysis capable of reducing the gradient size and runtime
- Blog entry

Clad Improvements

Petro Zarytskyi

Info

- Jan-ongoing 2024
- Restructured the system of storing and restoring original variables in the reverse mode. Simplified derivative statements and improved readability/performance.
- Major simplification in error estimation, which made possible by the new storing/restoring system. Improved performance and readability.
- Reimplemented jacobians using the vectorized forward mode. Improved the vectorized forward mode to prevent us from having regressions in jacobians.
- Replaced clad::array_ref in the derivative signature in favor of pointers. Replaced clad::array with C arrays.
- Introduced type cloning to handle variable arrays.
- Refactored GlobalStoreAndRef (store/restore inside loops), call expression differentiation, etc.
- Simplified the generated code: getting rid of all useless goto/label statements, introducing placeholder expressions to simplify multiplication differentiation results.
- Added support for new features: pointer-valued functions, pointer references, bitwise operators, basic cases of std::initializer_list, multiple indices in clad::gradient calls, etc.
- Small bug fixes: type safety, store/restore statement emission, etc.

Clad Integration in RooFit

Vaibhav Thakkar
Info

- Jan-Jun 2024
- Continued the work of Garima in RooFit where we flatten the compute graph and build a gradient for it
- Enabled large a large workflow from ATLAS open data
- Added support for computing only the hessian diagonal
- Implemented the differentiation graphs
- Many bug fixes and support work

Reverse-mode automatic differentiation of GPU (CUDA) kernels using Clad

Christina Koutsou

GSoC24, University of Thessaloniki, GR

Info

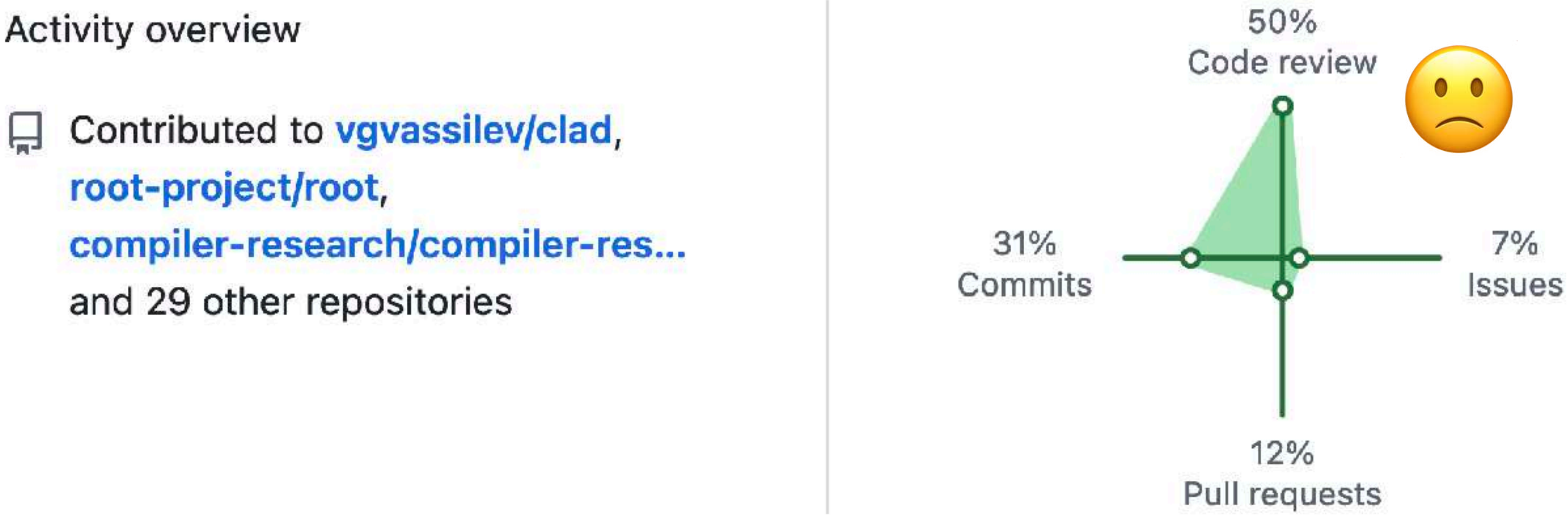
- May-ongoing 2024
- Enable CUDA support for both device and host functions
- Added CUDA builtins
- Enabled larger CUDA algorithms such as Black–Scholes
- Added write-race conditions synchronization primitives
- Added demos, benchmarks
- Improved documentation
- Blog entry

Running CR

Vassil Vassilev

Info

- Making sure all comes together.



Activity overview

Contributed to vgvassilev/clad, root-project/root, compiler-research/compiler-res... and 29 other repositories

How does that fit together?

Our mission is to conduct research in foundational software tools and adapt them for research in data science.

- We enabled Clad for large scale minimization fits in the field of High-Energy Physics. Demonstrated 10x improvement in minimization times for a single fit. Work is ongoing to make it available in flagship analysis tools such as CMS Combine. ATLAS is next.
- CppInterOp is being picked up for xeus-cpp, wasm, ROOT and Julia
- Clang-Repl is being adopted in ROOT through CppInterOp

How does that fit together?

- Made progress on making LLVM more robust on Windows
- Contributed to the LLVM community with infrastructure needs such as revamping the old website
- Continued to simplify cppyy using CppInterOp in efforts to connect both C++ and Python ecosystems
- Expanded to new frontiers in terms of agent-based simulations with BioDynaMo

Selected Papers

- Performance Portable Gradient Computations Using Source Transformation, accepted in 8th International Conference on Algorithmic Differentiation, September 16–20, 2024, Chicago Area, USA
- Optimization Using Pathwise Algorithmic Derivatives of Electromagnetic Shower Simulations, accepted in Computer Physics Communications

Selected Talks

- Automatic Differentiation of the Kokkos framework and the STL with Clad, 4th Mode Workshop
- Advanced optimizations for source transformation based automatic differentiation, 4th Mode Workshop
- Automatic Differentiation in RooFit for fast and accurate likelihood fits, ICHEP
- Taking derivatives of Geant4 - closer than you might think?, CHEP

Next Year Directions

- Increase the AD paper output
- Focus on AD non-HEP fields such as climate, floating point error estimation, ml.
- Further develop scientific cases for BioDynaMo and cppyy.
- Continue evolving our ecosystem

Next Meetings

- Monthly Meeting — 9th Jan, 1700 CET/0800 PDT

If you want to share your knowledge/experience with interactive C++ we can include presentations at an upcoming next meeting

Thank you!