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

Interactive C++: cling and clang-repl

Vassil Vassilev

11.11.2021
Status. Clang-Repl

- Added several tests for instantiating C++ templates on demand
Status. Clang-Repl. CppCon

- CppCon21 talk on Interactive C++ for Data Science and Differentiable Programming for C++
- 60 minute talks, good turnout
Status. Cling

- Continuing to rebase cling on top of llvm13
- xeus-clad is now done! Kudos to Ioana Ifrim and Chris Burr
Status. Clad in Xeus-Cling

---

AD Tutorial - CLAD & Jupyter Notebook

xeus- cling provides a Jupyter kernel for C++ with the help of the C++ interpreter cling and the native implementation of the Jupyter protocol xeus.

Within the xeus- cling framework, Clad can enable automatic differentiation (AD) such that users can automatically generate C++ code for the computation of derivatives of their functions.

```cpp
#include "clad/differentiator/Differentiator.h"
#include <iostream>

Forward Mode AD

For a function \( f \) of several inputs and single (scalar) output, forward mode AD can be used to compute (or, in case of Clad, create a function) computing a directional derivative of \( f \) with respect to a single specified input variable. Moreover, the generated derivative function has the same signature as the original function \( f \), however its return value is the value of the derivative.

```cpp
double fn(double x, double y) {
  return x*x + y*y;
}
```

```cpp
double fn_dx = clad::differentiate(fn, "x");
```

Reverse Mode AD

Reverse-mode AD enables the gradient computation within a single pass of the computation-graph of \( f \) using at most a constant factor (around 4) more arithmetical operations compared to the original function. While its constant factor and memory overhead is higher than that of the forward mode, it is
The document is ready. We are looking forward to your feedback.

Addressed several comments and still some minor improvements but mostly happy with the current state.
Status. Clad

- A talk by Ioana on “Automatic Differentiation for C++ and Cuda using Clad” at the 24th Euro AD Workshop
Status. 24th Euro AD Workshop Summary

- An event from 2nd to 4th of Nov (in the afternoons 1500 - 1900 CET)
- Theoretical and practical contributions to automatic differentiation
- AD tools and techniques spanning from OpenMP to C++ to Java to Julia
- The CERN Mode collaboration covered some of the physics-related ideas for AD
- Tribute to Andreas Griewank
The agenda is available here.

Many interesting use-cases.

S. Carli, W. Dekeyser, M. Blommaert and M. Baelmans
KU Leuven, Department of Mechanical Engineering

Yingkai Song Huiyi Cao Kamil A. Khan, Building AD-compatible linear underestimators of convex functions by sampling
The agenda is available [here](https://...).

Many interesting use-cases.

Shreyas Gaikwad, et al, An open-source tangent-linear and adjoint modeling framework for ice-sheet simulation enabled by the AD tool Tapenade

Giles Strong for the MODE Collaboration, TOMOPT: Differential Muon Tomography Optimisation
Plans

- Prepare a paper about the work we’ve completed.
- Enable error recovery for advanced C++ code (eg template instantiation)
- Accelerate upstreaming clang patches
- Automatically differentiate the CUDA kernels (including computation scheduler)
CaaS Open Projects

- Patches against clang.git
  - Implement FileManager uncaching
  - Adapt the user of invalidateCache to its new signature
  - Mark the file entry invalid, until reread
  - Propagate cache flags from LookupFile() to FileManager::getFile()
  - Pass the OpenFile flag also to DirectoryLookup
  - Do not load the source file just to get an irrelevant SourceLoc (ROOT-7111)
  - Allow interfaces to operate on in-memory buffers with no source location info [Pratyush Das]

- Open projects are tracked in our open projects page.
Next Meetings

- Monthly Meeting — 2nd December, 1700 CET/0800 PDT
- Tentative talk schedule:
  - LLDB, Raphael Isemann, Apple, Dec

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