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

Interactive C++: cling and clang-repl

Vassil Vassilev

01.06.2023

People

Vaibhav Thakkar

GSoC22, Electrical Engineering and Computer Science, Indian Institute of Technology, Kanpur, India

Implement vector mode in forward mode automatic differentiation in Clad. Project Info.

Sunho Kim

GSoC23, UCSD, CA, USA

Re-optimization using JITLink. Project Info.

Rishabh Bali

Unfunded contributor, B.Tech in Computer Engineering, Veermata Jijabai Technological Institute, Mumbai, India

Add support for differentiating with respect to multidimensional arrays (or pointers) in Clad. Project Info.

Anubhab Ghosh

GSoC23, Indian Institute of Information Technology, Kalyani, India

WebAssembly Support for clang-repl. Project Info.

People

Saqib

GSoD23, Pakistan

Improving the InterOp/Xeus-Clang-Repl documentation. Project Info.

Daemond Zhang

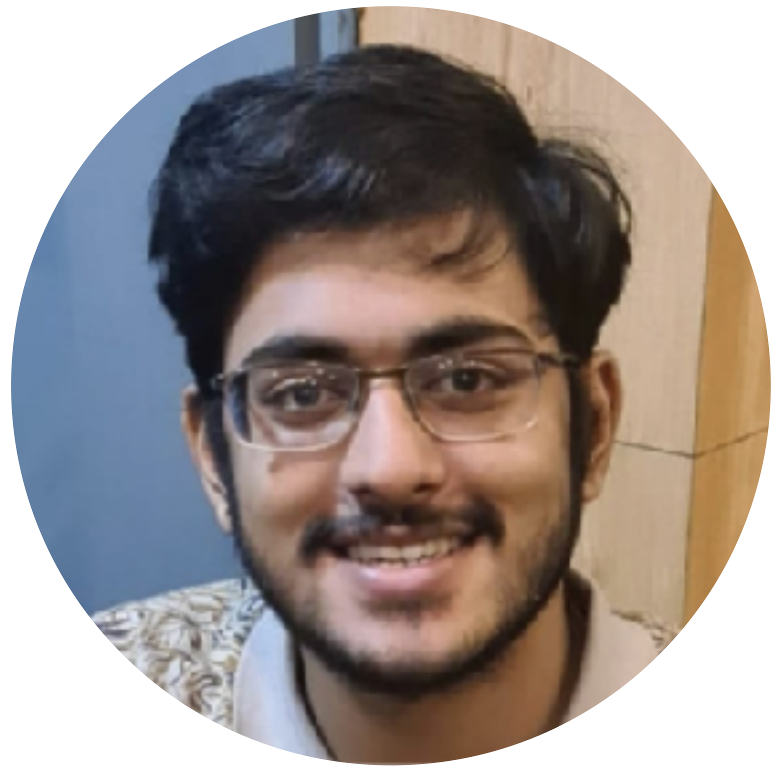
GSoC23, Tsinghua University, China

Improve automatic differentiation of object-oriented paradigms using Clad.Project Info.

Aaron Jomy

GSoC23, B. Tech in Computer Science, Manipal Institute of Technology, Manipal, India

Extend the Cppyy support in Numba. Project Info.



Krishna Narayanan

GSoC23, B.Tech in Electronics and Telecommunications, Veermata Jijabai Technological Institute, Mumbai, India

Tutorial development with clang-repl. Project Info.

Status. Cling

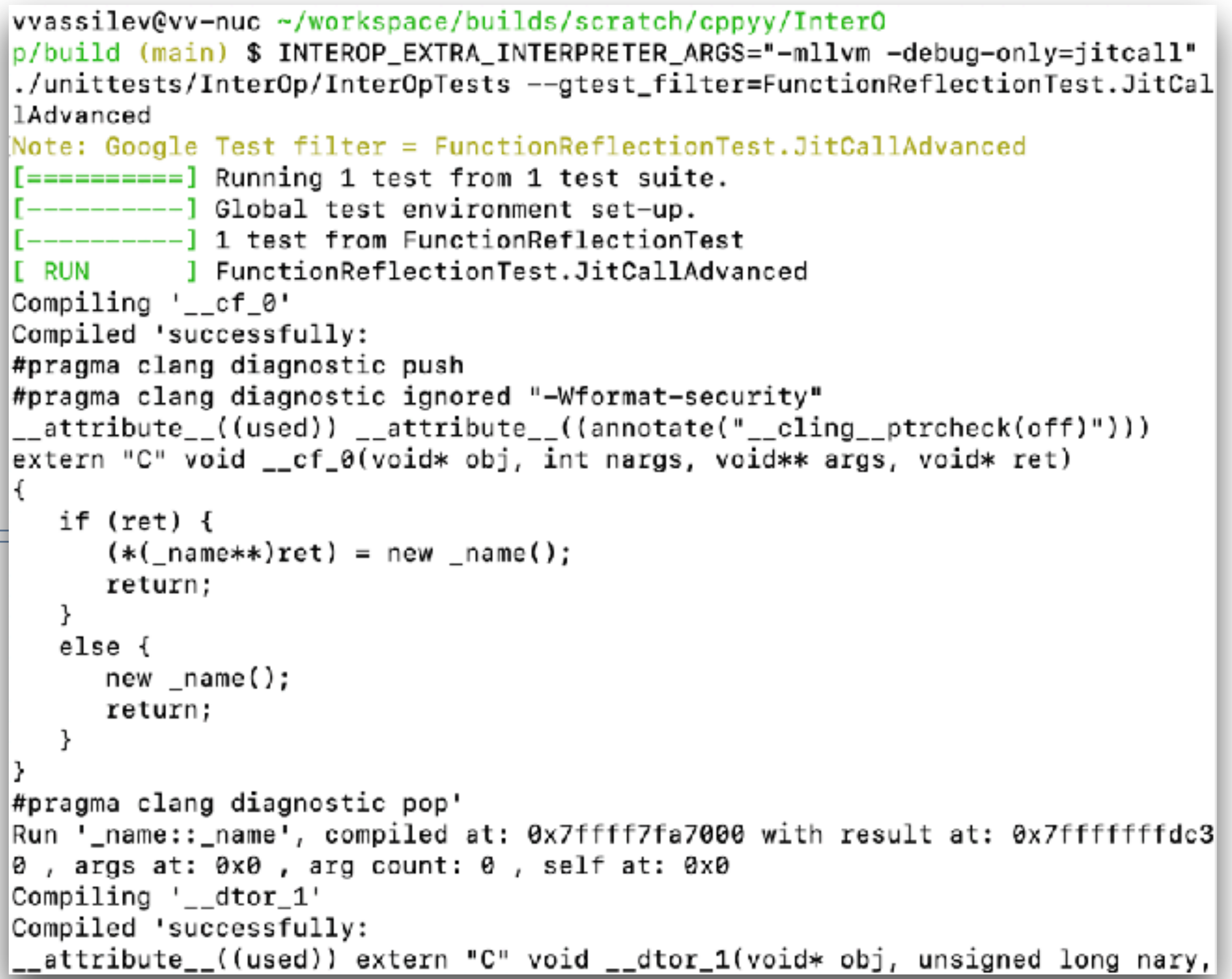
- Javier Lopez Gomez has a PR fixing all of the failures.

Status. Clang-Repl

- Incremental Input (RFC)
  - D143142 — Enable Lexer to grow its buffer
  - D143144 — Add TryGrowLexerBuffer/SourceFileGrower
  - D143148 — Add basic multiline input support
- Value Handling (RFC)
  - D146809 — [clang-repl] Implement Value pretty printing for containers
  - D141215 — Introduce Value and implement pretty printing. Quite far down the review process. **Landed!**
  - D146389 — Initial interactive CUDA support for clang-repl. Almost ready. **Landed!**

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

Status. InterOp



```
vvassilev@vv-nuc ~/workspace/builds/scratch/cppyy/InterO
p/build (main) $ INTEROP_EXTRA_INTERPRETER_ARGS="-mllvm -debug-only=jitcall"
./unittests/InterOp/InterOpTests --gtest_filter=FunctionReflectionTest.JitCal
lAdvanced
Note: Google Test filter = FunctionReflectionTest.JitCallAdvanced
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from FunctionReflectionTest
[ RUN      ] FunctionReflectionTest.JitCallAdvanced
Compiling '__cf_0'
Compiled 'successfully:
#pragma clang diagnostic push
#pragma clang diagnostic ignored "-Wformat-security"
__attribute__((used)) __attribute__((annotate("__cling__ptrcheck(off)")))
extern "C" void __cf_0(void* obj, int nargs, void** args, void* ret)
{
   if (ret) {
      (*(_name**)ret) = new _name();
      return;
   }
   else {
      new _name();
      return;
   }
}
#pragma clang diagnostic pop'
Run '_name::_name', compiled at: 0x7ffff7fa7000 with result at: 0x7fffffffdc3
0 , args at: 0x0 , arg count: 0 , self at: 0x0
Compiling '__dtor_1'
Compiled 'successfully:
__attribute__((used)) extern "C" void __dtor_1(void* obj, unsigned long nary,
```

- Completed the constructor/destructor support
- Removed the need to pass an interpreter pointer externally
- Improved the doxygen documentation
- Implemented a JitCall — a type-checked wrapper over CallFunc
- Implemented a flexible debug information printing in InterOp
- libInterOp-based cppyy: passes 185/504 tests.

Status. Clad

- Initial work on vector mode support

Status. Xeus-Clang-Repl/Xeus-Cpp

- Fixed issues with our binder setup

Upstreaming Patches

- Spreadsheet tracking the progress here.
- Total amount of upstreamed cling patches 26(26+0) out of 52 upstreamable.

CaaS Open Projects

- Open projects are tracked in our open projects page.

Next Meetings

- Monthly Meeting — 6th June, 1700 CET/0800 PDT

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

People

Jun Zhang

GSoC22, Contributor to Compiler Research

Optimize ROOT use of modules for large codebases, Upstream Value Printing in Clang-Repl and various improvements in LLVM/Clang.

Moving to industry for an internship

Thank you!