Project Proposal

CompResearch - Enable cross-talk between Python and C++ kernels in xeus-clang-REPL by using Cppyy

Organisation : CERN HSF
Mentor : Vassil Vassilev, Alexander Penev

Contact Information:
Full Name: Smit Parimal Shah
Degree: Bachelor of Technology in Computer Engineering (B. Tech)
University: Veermata Jijabai Technological Institute (VJTI) , Mumbai
Location: Mumbai, Maharashtra ,India
Time Zone: Indian Standard Time (UTC+5.30)
Email: ssmit1607@gmail.com, spshah_b21@ce.vjti.ac.in
Github Profile: @Smit1603
Resume: Link

1.1 About Me: Introduction & Background

Hello there! Smit Shah here. I am an undergraduate student in my second year at Veermata Jijabai Technological Institute (VJTI), Mumbai pursuing Computer Engineering as my major.

1. As for relevant courses in the institute, I have taken Data Interpretation and Analysis, Design and Analysis of Algorithms, Theory of Computations, Operating Systems, and Database Management Systems.
2. I have been developing projects and participating in various competitions since my first year.
    a. Image-Segmentation: Segment images using clustering and initialization algorithms. Link to its report.
    b. Spatial-AI: Developed and integrated a three-layer pipeline in OAK-D Pro to enhance depth estimation and disparity map generation capabilities.
3. I have decent experience with C++, Python, and git as demonstrated with projects and competitions. Also, Linux has been my daily driver for more than a year now and I am fairly comfortable with it, similarly, I use the terminal for most of my work.
4. I have been contributing to open source for quite a long time . Here are my some of the Open source contributions:
   InterOp : I have been exploring codebase and implemented number of TESTS for the same , following is the Pull Request for the same
    - https://github.com/compiler-research/InterOp/pull/32 (merged)
    - https://github.com/compiler-research/InterOp/pull/31 (merged)
    - https://github.com/compiler-research/InterOp/pull/29 (merged)
    - https://github.com/compiler-research/InterOp/pull/42 (merged)

1.2 Other Commitments

I can give 4 to 5 hours to the project everyday, compiling it to 30-35 hours per week or even more than that if required. I do not have any strict commitments to fulfill during this summer and can easily increase my duration if required. I am flexible with any working hours and can adjust them as per the time zone of my mentor. I have my summer vacation from May 31 to July 31 so a major part of the work will be easily completed in that duration. I will provide regular updates to my mentors regarding progress and will ask for their help whenever required. I'll continue to post updates on my blog every two weeks for reference.

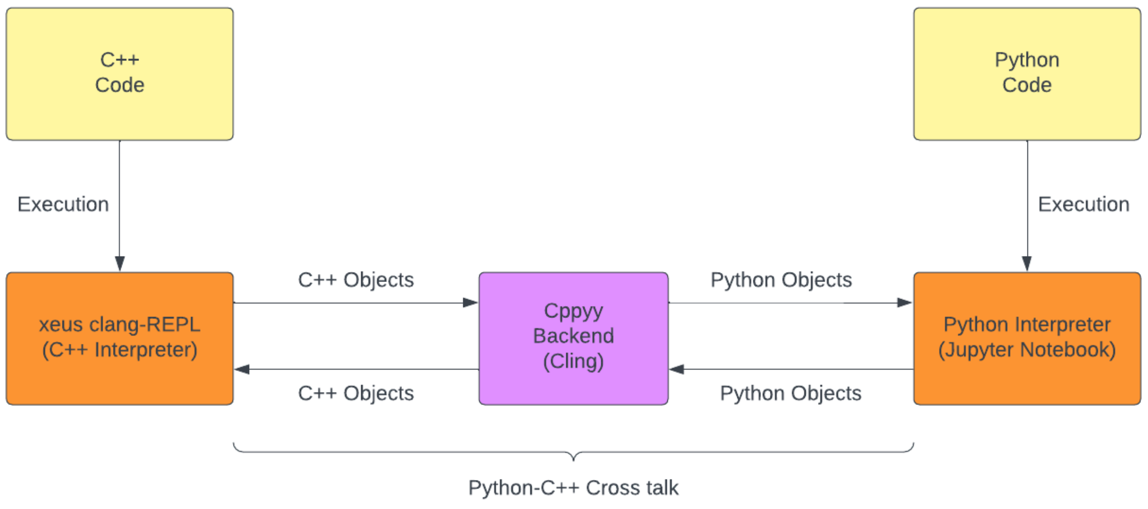
2.1 Project Description

The goal of this project is to allow for seamless communication between xeus-clang-REPL and Cppyy. This will allow users to write and execute C++ code inside a Jupyter notebook, while also being able to call C++ functions and use C++ classes from Python code within the same notebook.
Initially, cross-talk between xeus-clang-REPL and Cppyy was limited to passing primitive data types (such as integers and floats) between the two. However, this project aims to extend this cross-talk to support passing more complex data types, such as classes and functions, between C++ and Python.

First, let's break down the various components involved in this project:

> **Jupyter notebooks:**
> Jupyter notebooks are a popular web-based tool used for interactive computing. They allow users to create and share documents containing live code, equations, visualisations, and narrative text.
> **xeus-clang-REPL:**
> xeus-clang-REPL is a C++ kernel (using Clang-Repl) for Jupyter notebooks. It uses clang-REPL, which is a C++ interpreter, to execute C++ code inside a Jupyter notebook. Link to the github repo for more details.
> **Cppyy:**
> Cppyy is a Python-C++ binding generator that allows for seamless integration between C++ and Python. It generates Python bindings for C++ code, which allows Python code to call C++ functions and use C++ classes. Link for further details.

By allowing for seamless communication between C++ and Python within a Jupyter notebook, data analysts can take advantage of the strengths of each language. They can write their analysis pipeline in Python, which is known for its ease of use and rich ecosystem of data analysis libraries, while using C++ to execute computationally intensive tasks that require high performance. This can help reduce the time it takes to write and debug analysis pipelines, as well as improve overall performance.



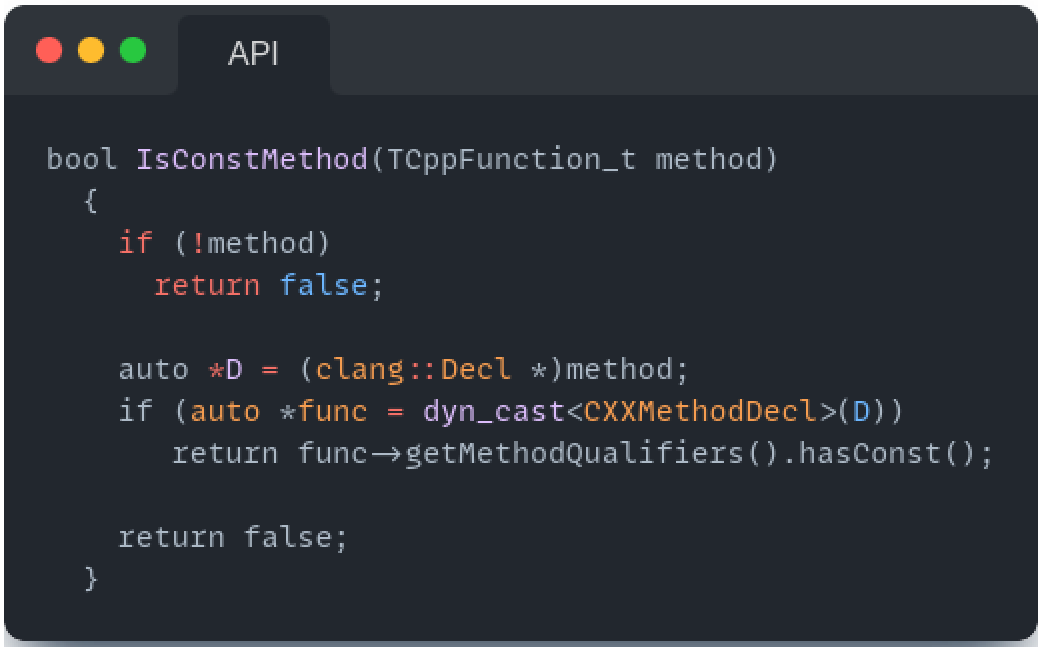
2.2 Milestones

- ➔ Task- [A] : Automate creation of equivalent Cppyy objects in the Python kernel when objects are created in the C++ kernel
- ➔ Task-[B] : Automate the creation of equivalent C++ objects in the xeus-clang-REPL kernel when Python objects are created
- ➔ Add documentation regarding the above implementation

2.3 Implementation: Plan of action

Basic Goal Of Project :
Teach InterOp and cppyy-backend to work with Clang-Repl supporting mixing Python and C++ through the Cppyy project. That includes changes on the repositories of InterOp, cppyy-backend, CPyCppyy and Cppyy. The overall metric is from currently 120 tests passing out of 498 to at least say 350 tests passing.

Step 1 : **Implementing APIs in InterOp library**
To implement APIs for various disabled tests as well as new tests as discussed with mentors. This APIs delivers required reflection information. Work will be similar to what we have done in 4 Pull Requests. Here is one of the snippets of API that have already been implemented and tested upon various test cases.
API snippet :

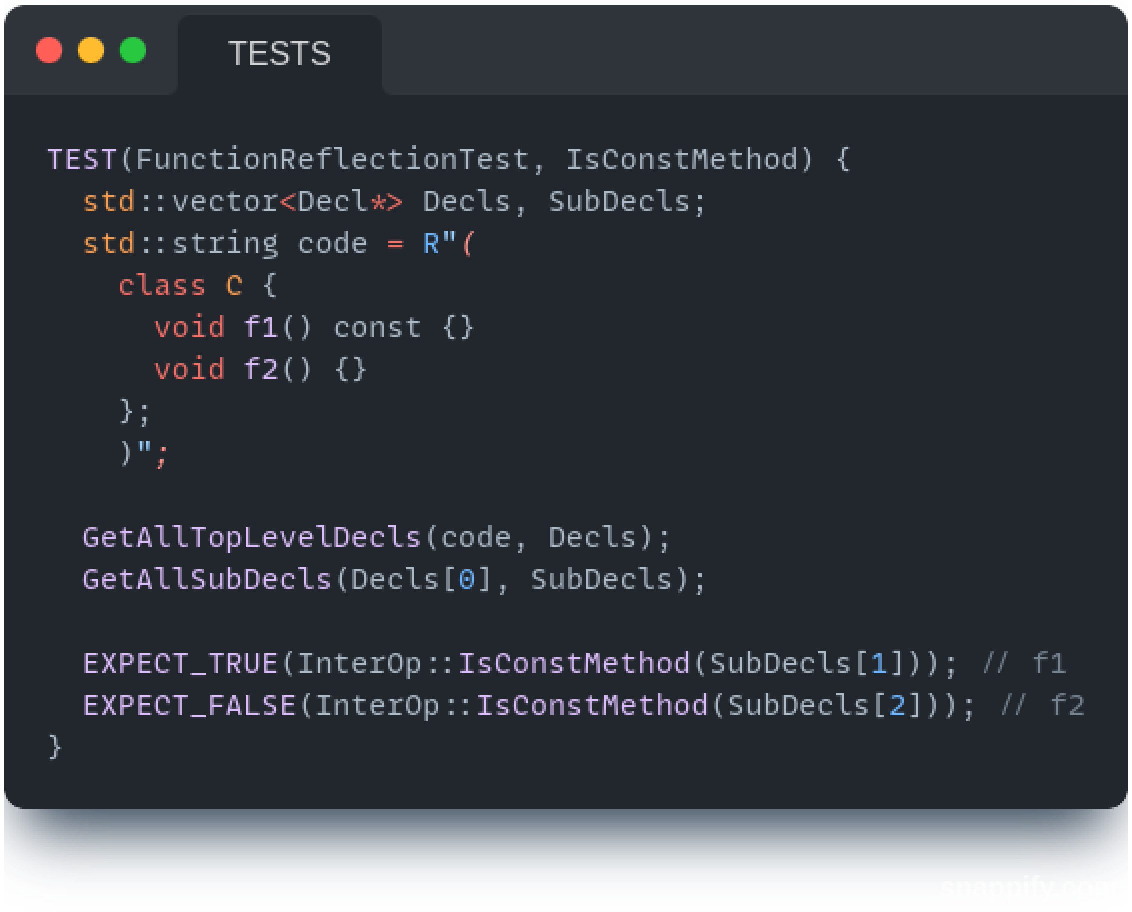

API

```
bool IsConstMethod(TCppFunction_t method)
  {
    if (!method)
      return false;

    auto *D = (clang::Decl *)method;
    if (auto *func = dyn_cast<CXXMethodDecl>(D))
       return func->getMethodQualifiers().hasConst();

    return false;
  }
```

Step 2 : **Implement Tests**
To test the above-implemented APIs for a variety of test cases to obtain high-quality output that completely passes all test cases. Earlier some test cases will be shared by mentors so that work is simplified but after due course of time, new test cases must be developed and checked for its viability.
TEST Snippet :


TESTS

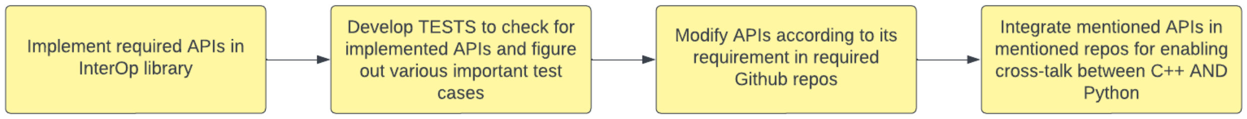
```
TEST(FunctionReflectionTest, IsConstMethod) {
  std::vector<Decl*> Decls, SubDecls;
  std::string code = R"(
    class C {
      void f1() const {}
      void f2() {}
    };
    )";

  GetAllTopLevelDecls(code, Decls);
  GetAllSubDecls(Decls[0], SubDecls);

  EXPECT_TRUE(InterOp::IsConstMethod(SubDecls[1])); // f1
  EXPECT_FALSE(InterOp::IsConstMethod(SubDecls[2])); // f2
}
```

Step 3 : **Modify APIs as well as Test Cases according to requirements in other repository**
Modify all implemented APIs and Tests according to their requirements and usage in other repositories like cppyy-backend and CPyCppyy. This will help in easy binding of Python and C++ thus enabling efficient cross-talk.

Step 4 : **Integrating above implemented APIs in mentioned repositories**
Understand the codebase of above mentioned repositories and integrate APIs in respective repositories so that the main objective of our project is attained and cross-talk is enabled. There should be proper syncing of APIs with backend repos for efficient and easy cross-talk.


Implement required APIs in InterOp library
Develop TESTS to check for implemented APIs and figure out various important test cases
Modify APIs according to its requirement in required Github repos
Integrate mentioned APIs in mentioned repos for enabling cross-talk between C++ AND Python

2.4 Project Timelines

Up Till May 4	**Proposal accepted or rejected** ➔ Familiarise with codebase ➔ Understand the basics of Clang/Cling which is highly required for designing plugins and mapping functions ➔ Discuss with the mentor about the doubts related to implementation. ➔ A proper project structure will ensure great development. **Deliverables :** Have good and adequate knowledge about Clang/Cling for easy implementation of projects. Have clear ideas and requirements of the project with no doubts.
May 4 - May 28	**Community Bonding Period** ➔ Set up InterOp, cppyy-backend, CPyCppyy and Cppyy environment as specified in the github continuous integration recipe. ➔ Start working on the reflection function in InterOpDeliverables : Implement APIs and try to think of some new Test Cases for improvised implementation. **Deliverables :** Add 10 failing TESTS in InterOp and implement 5 PR with fixes for Tests in InterOp with some extra test cases.
Week 1	**Coding officially begins!** ➔ Build Python in Debug mode and show that we can run the gdb debugger on a failing test in Cppyy. **Deliverables :** To build Python in debug mode
Week 2	➔ Start working on Step 1 of implementation ➔ Start working on APIs to be implemented after discussing with mentors ➔ Try to find best and efficient method for developing APIs **Deliverables :** Implement required APIs by discussing with mentors

Week 3	➔ Test all developed APIs with various test cases mentioned by mentors. **Deliverables :** Test all implemented APIs
Week 4	➔ Try to find distinguishing test cases along with FIXMEs for many APIs for improving APIs use case and efficiency. **Deliverables :** Build familiar knowledge to develop own distinguishing test cases to test APIs
Week 5	**Buffer Period** ➔ Try to complete any task mentioned before if not completed or any unforeseen situation arises ➔ Coordinate with mentors, whether any new task is to be implemented.
Week 6 - Week 7	➔ Modify APIs according to their requirement in Cppyy-backend and CPyCppyy. ➔ Codebase of both repo must be acquainted with. **Deliverables :** Modify APIs and understand the codebase of both repositories for easy integration in future.
Week 8 - Week 9	➔ Integrate developed APIs in backend repos for binding Python with C++ ➔ This enables cross-talk between Python and C++ ➔ Attain proper syncing with backend **Deliverables :** Integrate all required APIs efficiently to enable cross-talk functionality required
Week 10 - Week 11	**Buffer Period** ➔ Try to complete any task remaining or any easily implementable stretch goals of this project. ➔ Try to fulfil all necessary requirements of mentors regarding the project.
Week 12	➔ Work on documentation along with blog post
Week 13	➔ Summarise all the work in a blog post and submit it for evaluation.

3. Why me and my Motivation?

Low level details of computer systems have always intrigued me . I have been interested in system programming and compilers . Also while solving various tasks/issues my interest level increased manifold . I have always aspired to work for CERN and will be highly grateful to receive the opportunity to contribute as a Google Summer of Code student developer. This is a great opportunity and a head start for achieving success in my life.

4. References

- Clang, LLVM, Cling Reference Documentation
- Cppyy Reference Documentation
- clang-REPL Plugin Development
- https://github.com/compiler-research/InterOp
- https://github.com/compiler-research/cppyy-backend
- https://github.com/compiler-research/CPyCppyy