GSoC 2023
Program @ LLVM

Student:
Krishna Narayanan

Mentors:
Vassil Vassilev, David Lange

Tutorial Development with clang-repl and xeus-clang-repl

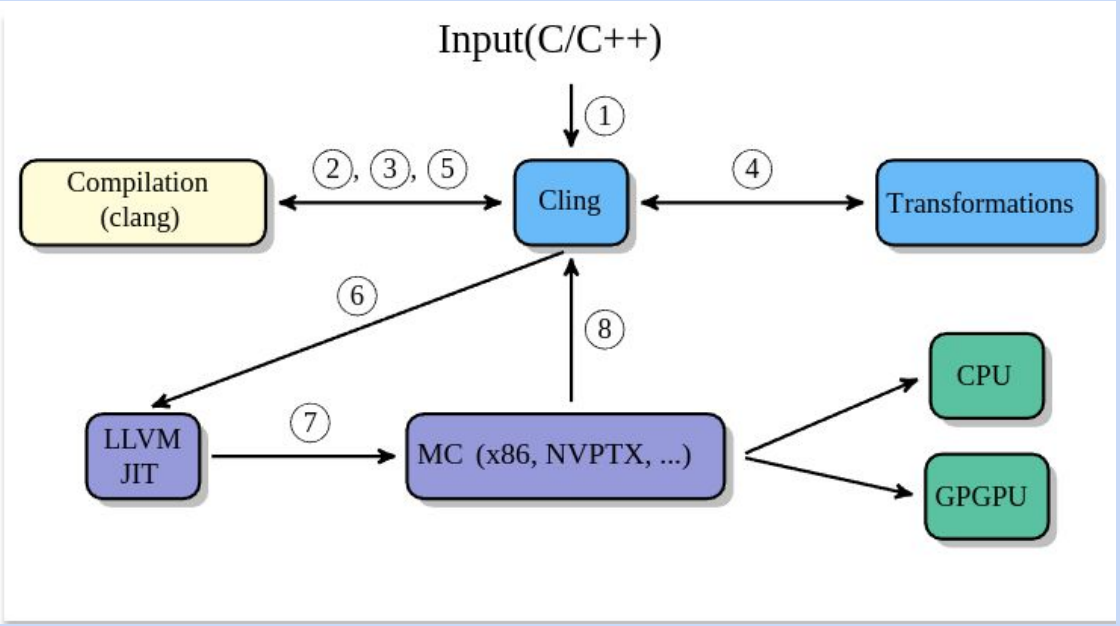
Incremental Compilation

- A Read-Eval-Print Loop, or REPL, is an environment where user inputs are read and evaluated, and then the results are returned to the user.
- REPLs provide an interactive environment to explore tools available in specific environments or programming languages

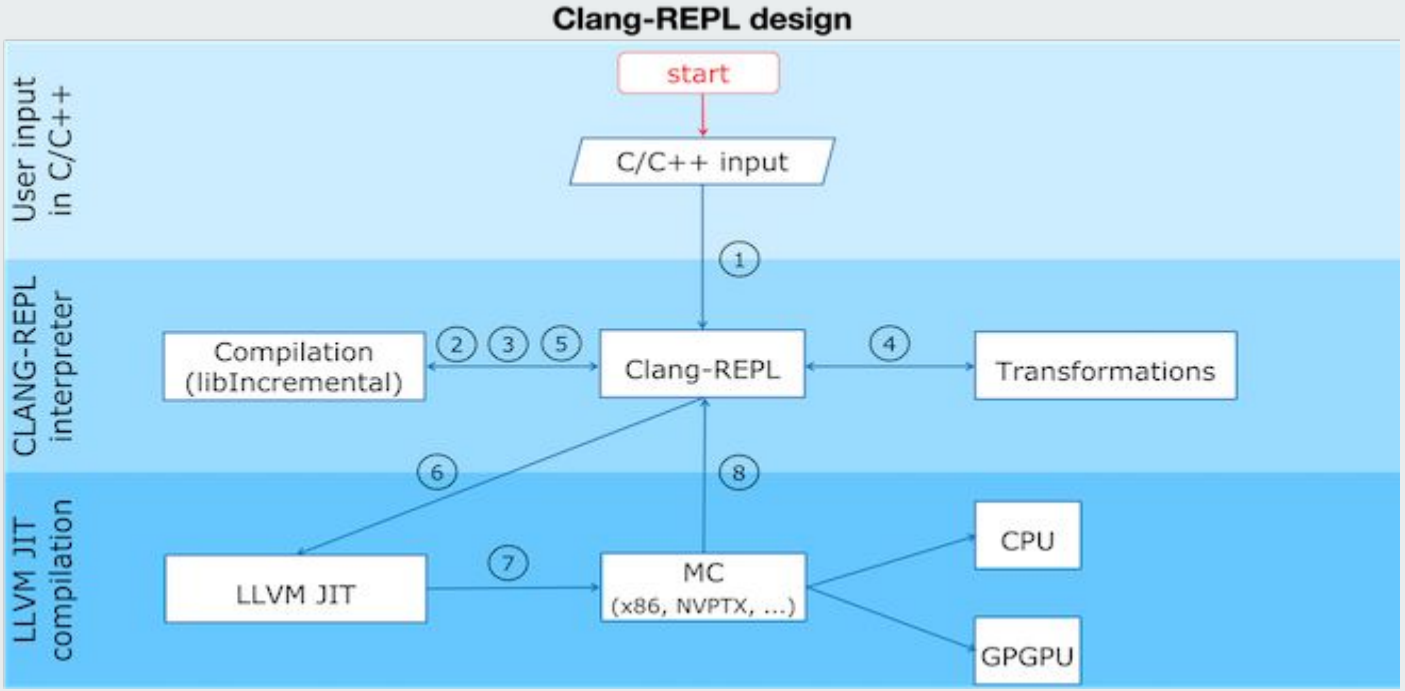
```
krishna@krishna:~$ expr World!
World!
krishna@krishna:~$ expr 10/2
10/2
krishna@krishna:~$ expr 10 / 2
5
krishna@krishna:~$
```

```
[cling]$ #include<iostream>
[cling]$ int sum(int a, int b) { int c = a+b; std::cout << c << std::endl; return c+1;}
[cling]$ int c  = sum (8,9)
17
(int) 18
[cling]$
```

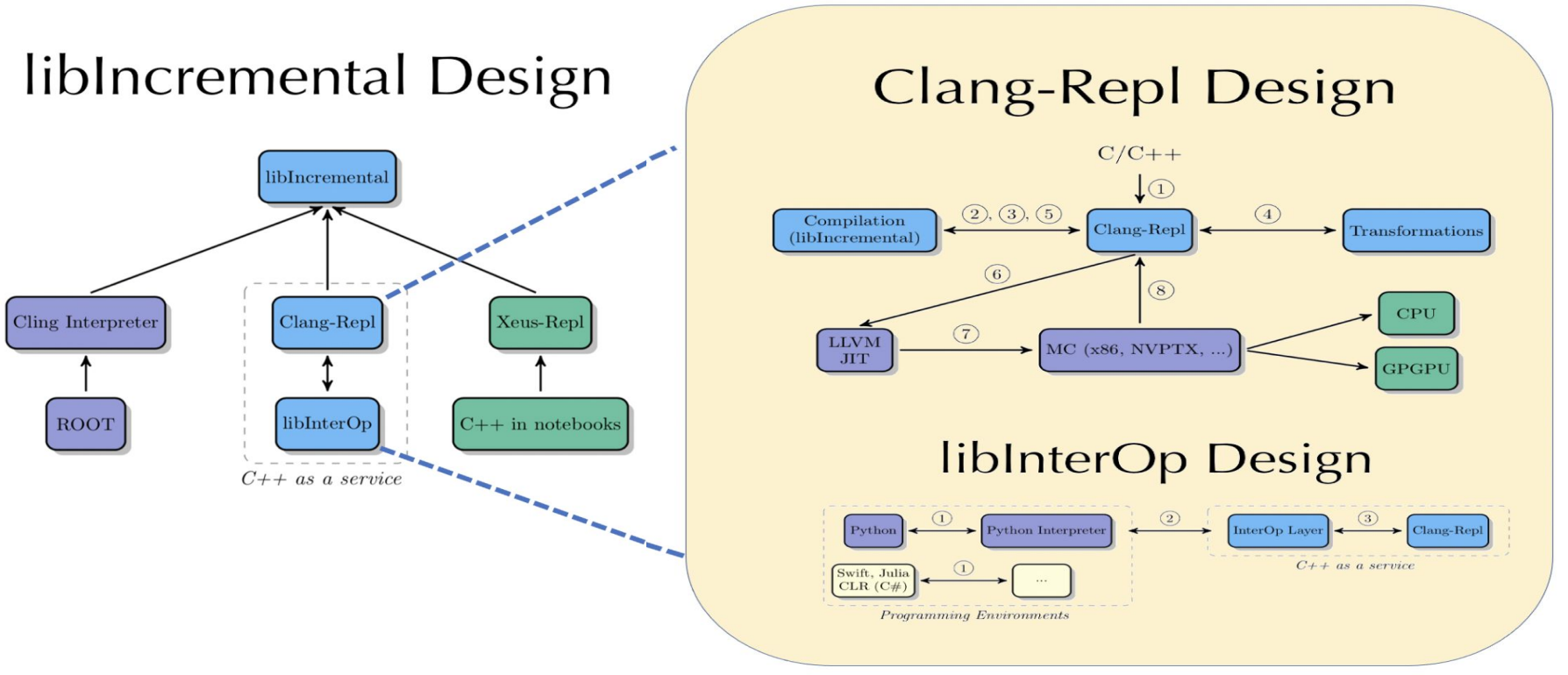
Cling


Input(C/C++)
1
Compilation
(clang)
2, 3, 5
Cling
4
Transformations
6
8
LLVM
JIT
7
MC (x86, NVPTX, ...)
CPU
GPGPU

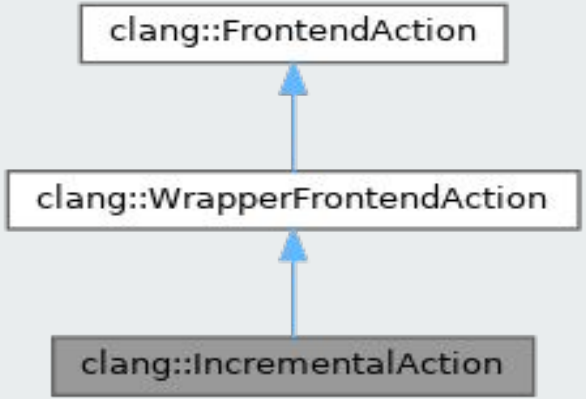
Clang-REPL


Clang-REPL design
User input in C/C++
CLANG-REPL interpreter
LLVM JIT compilation
start
C/C++ input
1
Compilation (libIncremental)
2
3
5
Clang-REPL
4
Transformations
6
8
LLVM JIT
7
MC (x86, NVPTX, ...)
CPU
GPGPU

Overview


libIncremental Design
libIncremental
Cling Interpreter
Clang-Repl
Xeus-Repl
ROOT
libInterOp
C++ in notebooks
C++ as a service
Clang-Repl Design
C/C++
1
Compilation (libIncremental)
2, 3, 5
Clang-Repl
4
Transformations
6
8
LLVM JIT
7
MC (x86, NVPTX, ...)
CPU
GPGPU
libInterOp Design
Python
1
Python Interpreter
2
InterOp Layer
3
Clang-Repl
C++ as a service
Swift, Julia CLR (C#)
1
...
Programming Environments

What makes the difference between **CLING and Clang-REPL**



Examples

DEMOS:

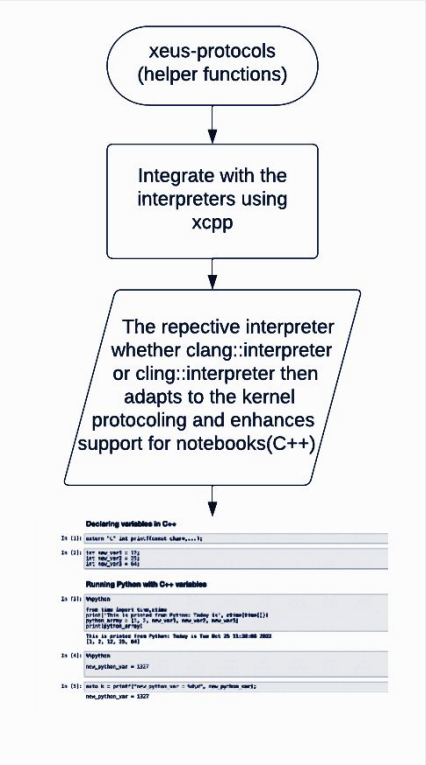
CLING : https://asciinema.org/a/BQxbyakFSOkife64hSKvU223y

CLANG-REPL: https://asciinema.org/a/MCW63Tqvy1hPQrVNr1zdWdU18

```
[cling]$ int a = 10;
[cling]$ extern int a;
[cling]$ static int a;
input_line_5:2:13: error: redefinition of 'a'
 static int a;
            ^
input_line_3:2:6: note: previous definition is here
 int a = 10;
     ^
[cling]$
```

```
clang-repl> int a = 10;
clang-repl> extern int a;
clang-repl> static int a ;
In file included from <<< inputs >>>:1:
input_line_2:1:12: error: static declaration of 'a' follows non-static declaration
static int a ;
           ^
input_line_1:1:12: note: previous declaration is here
extern int a;
           ^
error: Parsing failed.
```

Xeus-protocols




```
nl::json execute_request_impl(int execution_counter,
                              const std::string& code,
                              bool silent,
                              bool store_history,
                              const nl::json::node_type* user_expressions,
                              bool allow_stdin) override;

nl::json complete_request_impl(const std::string& code,
                               int cursor_pos) override;

nl::json inspect_request_impl(const std::string& code,
                              int cursor_pos,
                              int detail_level) override;

nl::json is_complete_request_impl(const std::string& code) override;

nl::json kernel_info_request_impl() override;
```

Adapting to xeus protocols

xeus-clang-repl

```
namespace xcpp {
class XEUS_CLANG_REPL_API interpreter : public xeus::xinterpreter {
public:
  interpreter(int argc, const char *const *argv);
  virtual ~interpreter();
```

```
std::unique_ptr<clang::Interpreter> m_interpreter;
```

xeus-cling

```
namespace xcpp
{
    class XEUS_CLING_API interpreter : public xeus::xinterpreter
    {
    public:

        interpreter(int argc, const char* const* argv);
        virtual ~interpreter();
```

```
cling::Interpreter m_interpreter;
cling::InputValidator m_input_validator;
```

Our Aim !

Declaring variables in C++

In [1]:
```
extern "C" int printf(const char*,...);
```

In [2]:
```
int new_var1 = 12;
int new_var2 = 25;
int new_var3 = 64;
```

Running Python with C++ variables

In [3]:
```
%%python

from time import time,ctime
print('This is printed from Python: Today is', ctime(time()))
python_array = [1, 2, new_var1, new_var2, new_var3]
print(python_array)
```

```
This is printed from Python: Today is Tue Oct 25 11:38:08 2022
[1, 2, 12, 25, 64]
```

In [4]:
```
%%python

new_python_var = 1327
```

In [5]:
```
auto k = printf("new_python_var = %d\n", new_python_var);
```

```
new_python_var = 1327
```

Any Questions !?

THANK YOU !