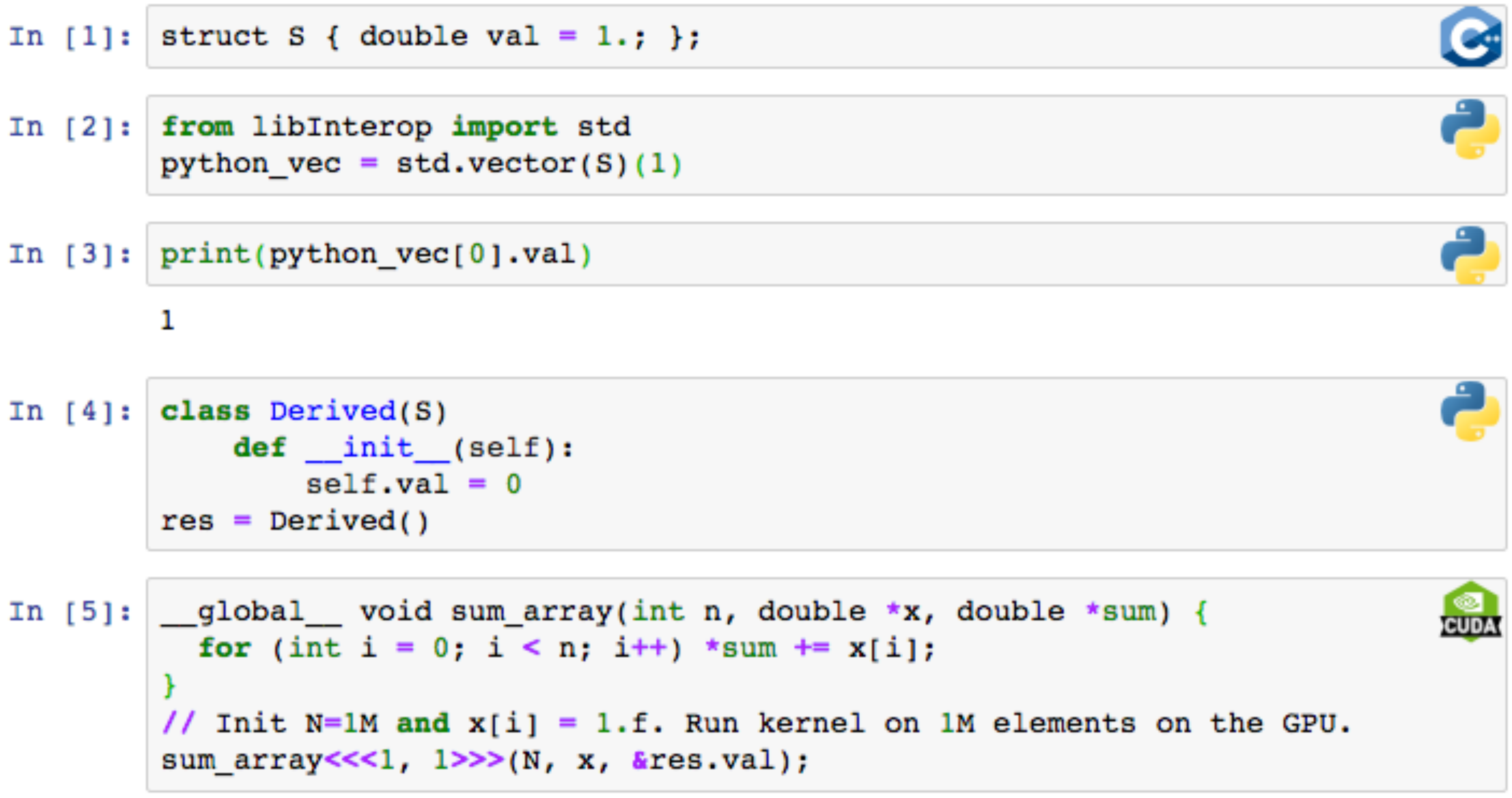
The CaaS Project. Progress & Plans Q1, Q2

Vassil Vassilev

Project Goals

- Support for incremental compilation (clang::libInterpreter, Clang-Repl)
- Language interoperability layer (cppyy, libInterOp)
- Heterogeneous hardware support (offload execution, clad demonstrator)
- Use case development & community outreach (tutorial development, demonstrators)

Project Goals



```
In [1]: struct S { double val = 1.; };
```

```
In [2]: from libInterop import std
        python_vec = std.vector(S)(1)
```

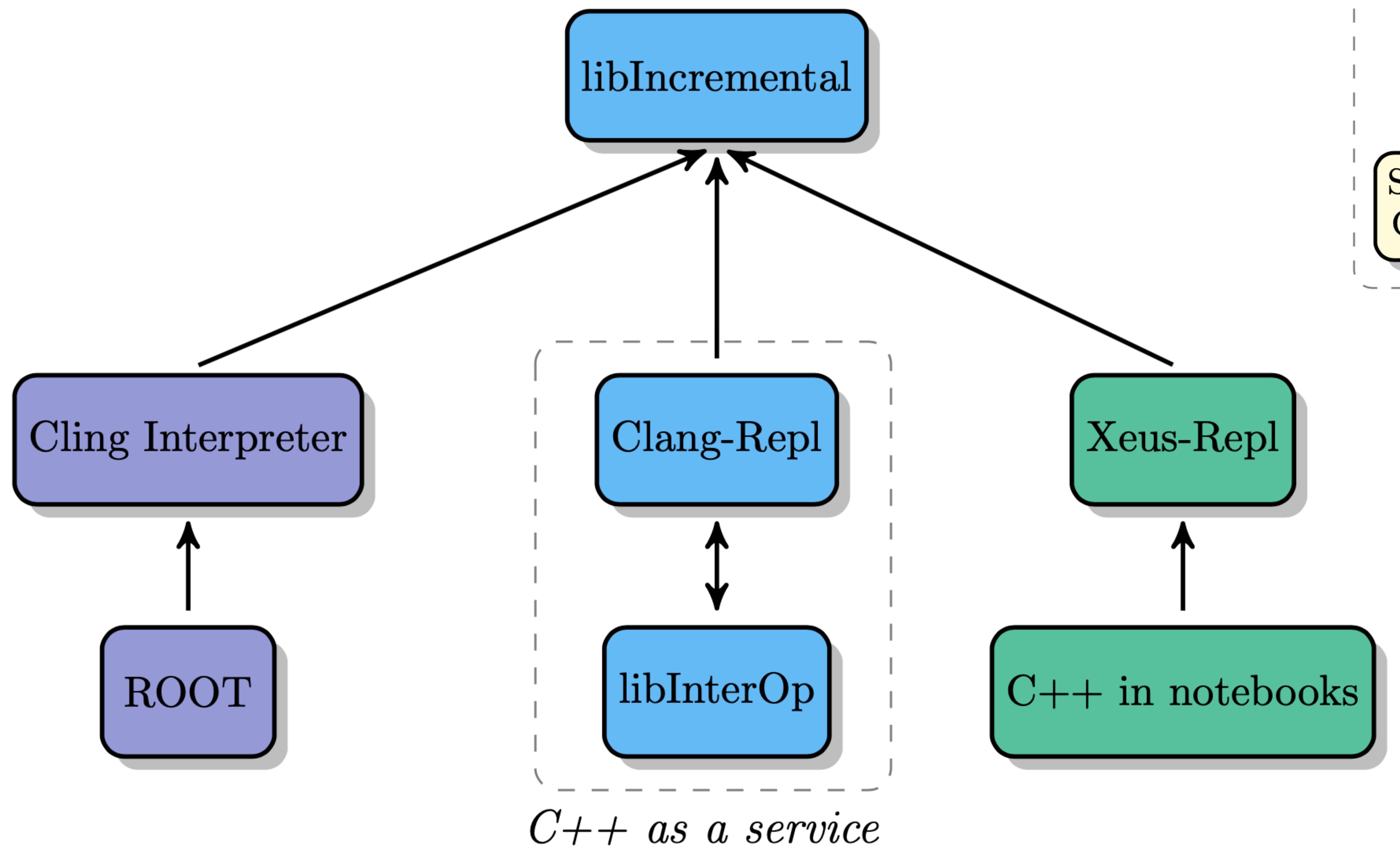
```
In [3]: print(python_vec[0].val)

        1
```

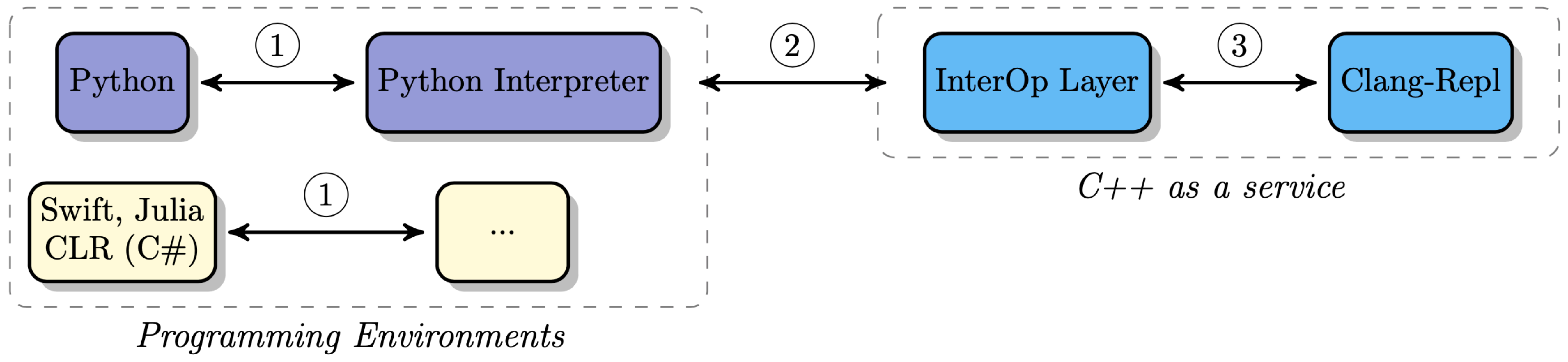
```
In [4]: class Derived(S)
            def __init__(self):
                self.val = 0
        res = Derived()
```

```
In [5]: __global__ void sum_array(int n, double *x, double *sum) {
          for (int i = 0; i < n; i++) *sum += x[i];
        }
        // Init N=1M and x[i] = 1.f. Run kernel on 1M elements on the GPU.
        sum_array<<<1, 1>>>(N, x, &res.val);
```

Enable bi-directional language communication capable of controlling accelerator hardware

Project Goals





Reroute the cling-based ecosystem more to llvm upstream

Q1 Progress

1. [Q1/VV] Upgrade to LLVM 13 — 90% complete

2. [Q1/VV] Update Cling to use more of LLVM13 — 60% complete (depends on 1.)

3. [Q1/DL] Construct simple patches to upstream dashboard to track — 100% complete

4. [Q1-Q4/VV, GS, BK] Upstream Cling-specific patches — 0% complete

5. [Q1-Q4/DL] Keep track of Cling SLoC — Q1
 50 files changed, 695 insertions(+), 1167 deletions(-)

6. [Q1/II] ACAT proceedings — sent for review

7. [Q1/PA] Support Tensors and showcase differentiation of Eigen entities — 50% complete

8.[Q1/GS] Deliver error estimation talk at SIAM incl the req. development— complete

Carry-over for Q2

1. Connect Clang-Repl to the Python Interpreter — Q1/BK → Q2/II
   The python interpreter provides C API which allows to expose itself and switch to writing python code on the prompt. In ROOT this happens via TPython::Prompt and we want the modern version of this for clang-repl.

2. Rebase cppyy to use cling-only interfaces (making cppyy ROOT-independent) — Q1/BK → Q2
   The task is about transforming the various ROOT Meta layer calls to their underlying clang/cling analogs

3. Define a set of new classes which handle what's needed (eg TClingCallFunc, etc) — Q1/BK → Q2
   The task is about extracting the common cases where we need a lot of boilerplate code and provide abstractions for it. For example, the mechanism to call functions in a uniform way (currently done with TClingCallFunc) needs to modernized into its own ROOT-independent entity in libInterOp

Carry-over for Q2

9. Connect libInterOp with clang-repl (see 6)— Q1/BK → Q2
   The python interpreter provides C API which allows to expose itself and switch to writing python code on the prompt. In ROOT this happens via TPython::Prompt and we want the modern version of this for clang-repl.

10. Improve test cases and demonstrators — Q1/II → Q2
    The task is about updating the existing demonstrators and developing new ones given the advances in Clad.

11. Differentiate CUDA kernels — Q1/II → Q2/PA?

12. Support Tensors and showcase differentiation of Eigen entities — Q1/PA → Q2

Plans for Q2

15. Implement in clang an extension to allow statements on the global scope — Q2/VV

16. Add extensible value printing facility — Q2/VV

17. Advance error recovery and code unloading — Q2/PC
    The task is to make clang-repl more robust when it comes to surviving from errors.

18. Design and Develop a CUDA engine working along with C++ mode —Q2/II,SSP
    The task is to improve and generalize the implementation of the PTX support in cling and demonstrate it in clang-repl.

19. Rebase cppyy to use clang-repl/libInterpreter interfaces — Q2/BK

20. Develop demonstrators (eg the one from the Jupiter mockup) — Q2/BK

21. Design and implement a backend capable of offloading computations to a GPGPU. Assess technical performance of gradient produced by Clad on GPGPU — Q2/II,VV

Plans for Q2

22. Add more clad benchmarks — Q2/DL

23. Add extensible value printing facility — Q2/VV → Q3

24. Write a paper on incremental C++ — Q2/VV

25. Write a paper on AD for the aggregate types — Q2/PA

26. Write an Error Estimation paper — Q2/GS

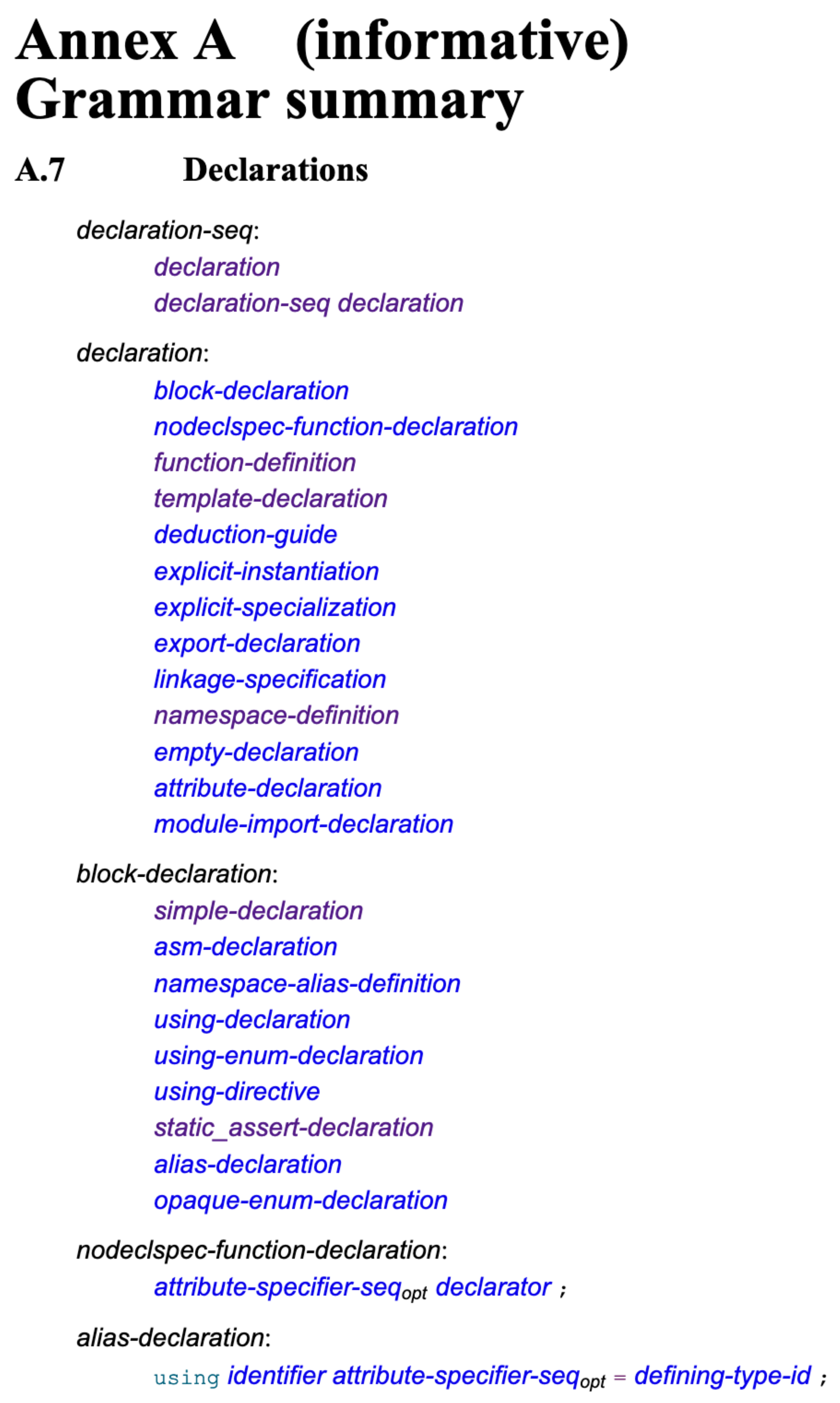
Extra Contributors

- Quite a bit of interest this year through IRIS-HEP, GSoC, GDOC and unfunded contributors
- More experienced people should help mentoring
- More clarity after May 20.

Implement in clang an extension to allow statements on the global scope

Extending C++

- Extend the clang parser when building incremental TU
- Figure out which is a good place in the grammar to extend. (Mind CUDA, attribute parsing vs non-attributes)
- Model the new statements in the PartialTranslationUnit (Mind static init and performance)



Annex A (informative)
Grammar summary

A.7 Declarations

declaration-seq:
    declaration
    declaration-seq declaration

declaration:
    block-declaration
    nodeclspec-function-declaration
    function-definition
    template-declaration
    deduction-guide
    explicit-instantiation
    explicit-specialization
    export-declaration
    linkage-specification
    namespace-definition
    empty-declaration
    attribute-declaration
    module-import-declaration

block-declaration:
    simple-declaration
    asm-declaration
    namespace-alias-definition
    using-declaration
    using-enum-declaration
    using-directive
    static_assert-declaration
    alias-declaration
    opaque-enum-declaration

nodeclspec-function-declaration:
    attribute-specifier-seq$_{opt}$ *declarator* ;

alias-declaration:
    `using` *identifier attribute-specifier-seq*$_{opt}$ = *defining-type-id* ;