Cling Updates

Vassil Vassilev, Princeton

https://compiler-research.org

Cling Overview

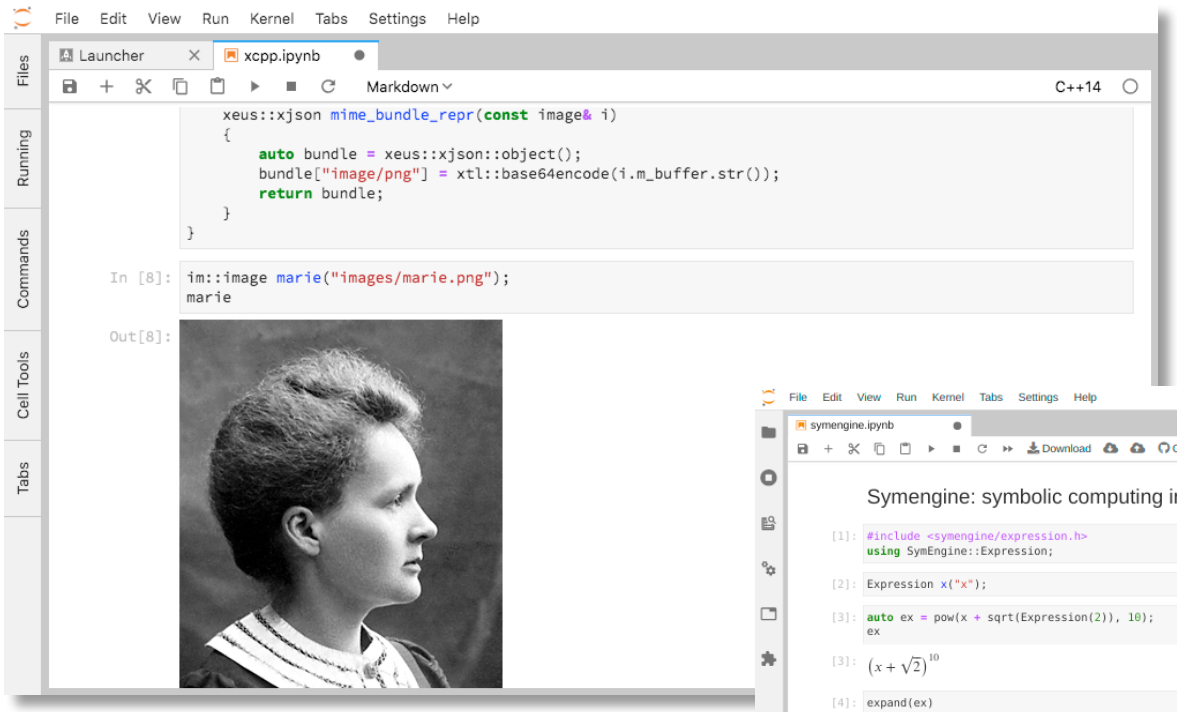
- The LLVM-based C++ interpreter used in ROOT6
- Enables interactivity, IO, PyROOT/cppyy
- Can be used as a standalone tool
- The C++ engine behind interactive C++ with Jupyter via zeus-cling

```
[root] ntuple->GetTitle()
error: use of undeclared identifier 'ntuple'
[root] TFile::Open("tutorials/hsimple.root"); ntuple->GetTitle()
(const char *) "Demo ntuple"
[root] gFile->ls();
TFile**         tutorials/hsimple.root    Demo ROOT file with histograms
 TFile*         tutorials/hsimple.root    Demo ROOT file with histograms
  OBJ: TH1F     hpx    This is the px distribution : 0 at: 0x7fadbb84e390
  OBJ: TNtuple    ntuple    Demo ntuple : 0 at: 0x7fadbb93a890
  KEY: TH1F     hpx;1    This is the px distribution
  [...]
  KEY: TNtuple    ntuple;1    Demo ntuple
[root] hpx->Draw()c
```

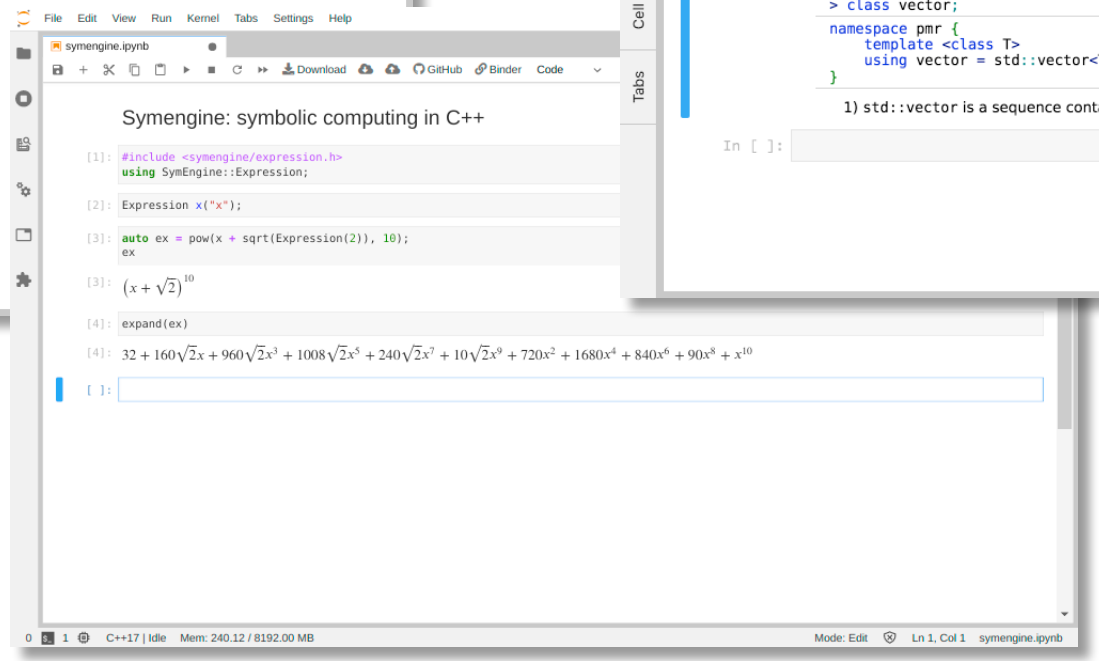
Four Cling Releases Since 2018

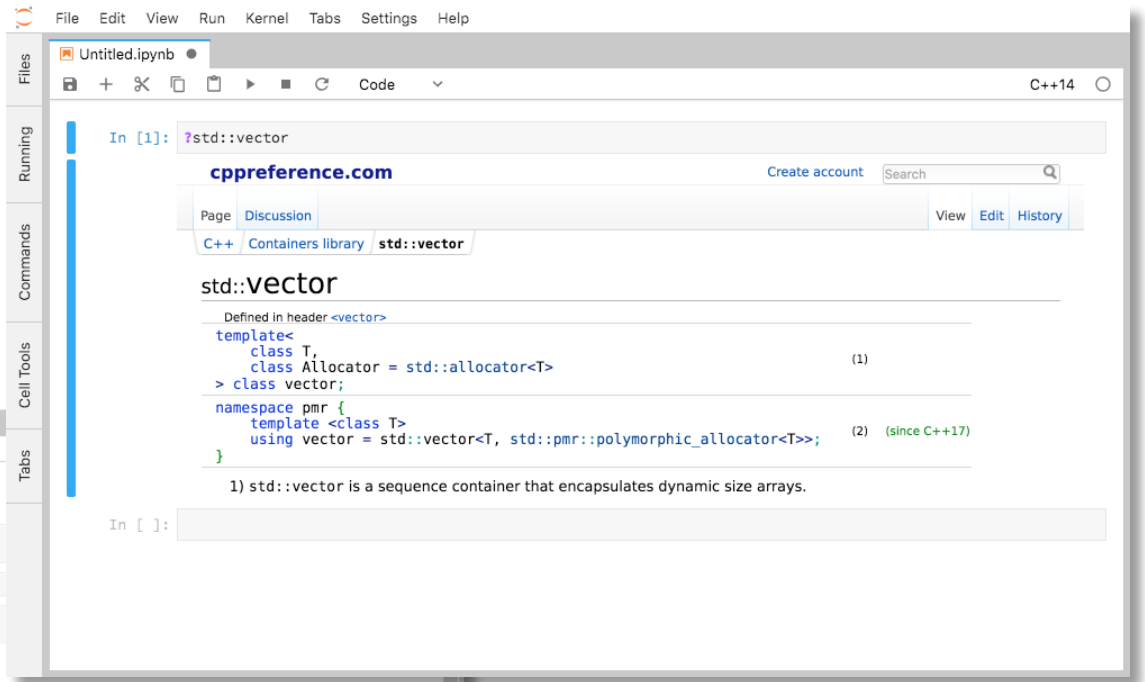
- v0.6
  - Enables Plugins
  - Adds automatic differentiations support with clad
  - Emulated thread local storage
- v0.7
  - Implement a mechanism allowing to redefine entities with the same name
  - CUDA support
  - Symbol resolver based on the binary object formats
- v0.8
  - Complete the C++ Modules used in ROOT's dictionaries
- v0.9
  - Upgrade to LLVM9
  - Improvements in the CUDA backend and
  - Improvements in C++ Module support
  - Reduced dependence on custom patches

Xeus-Cling. C++ in Notebooks



Visualization of user-defined images



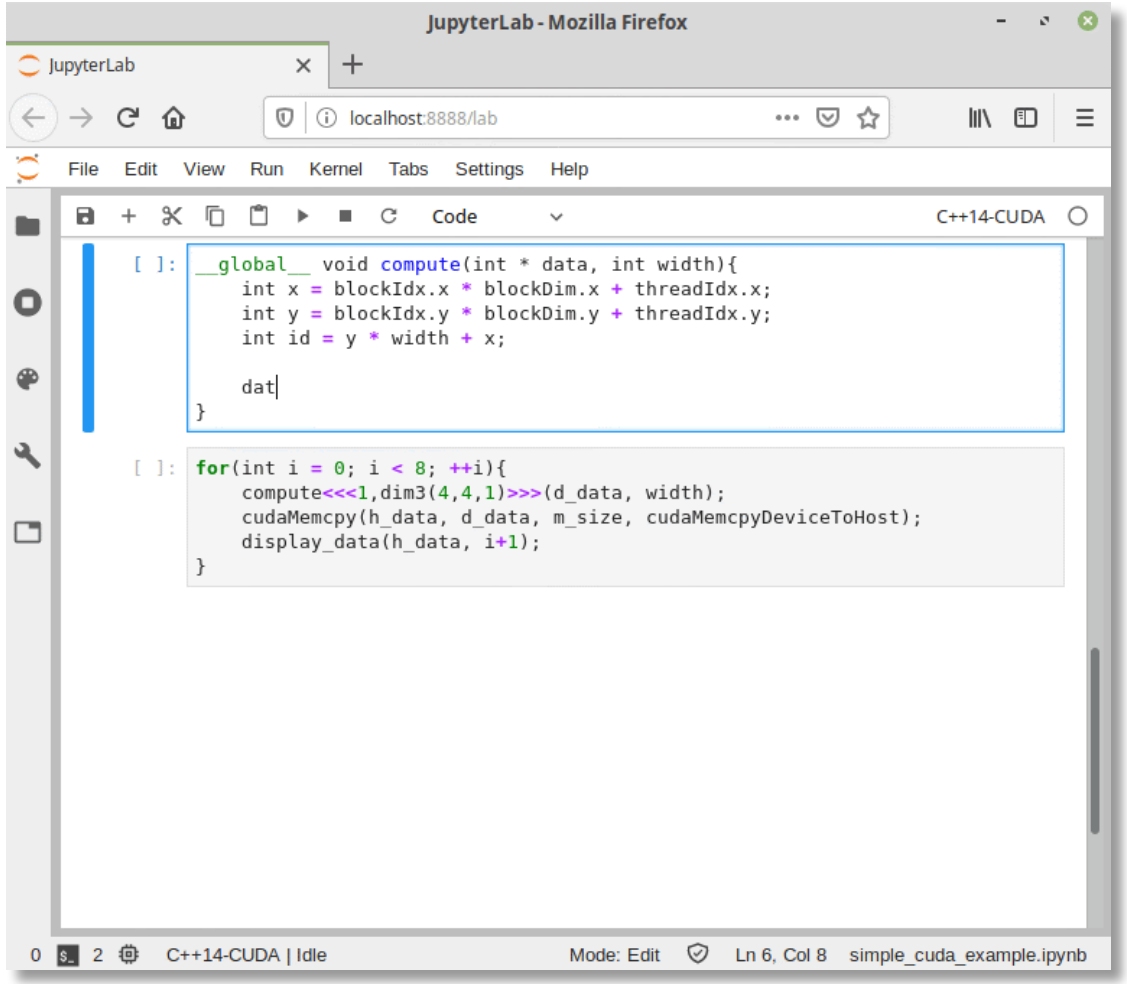


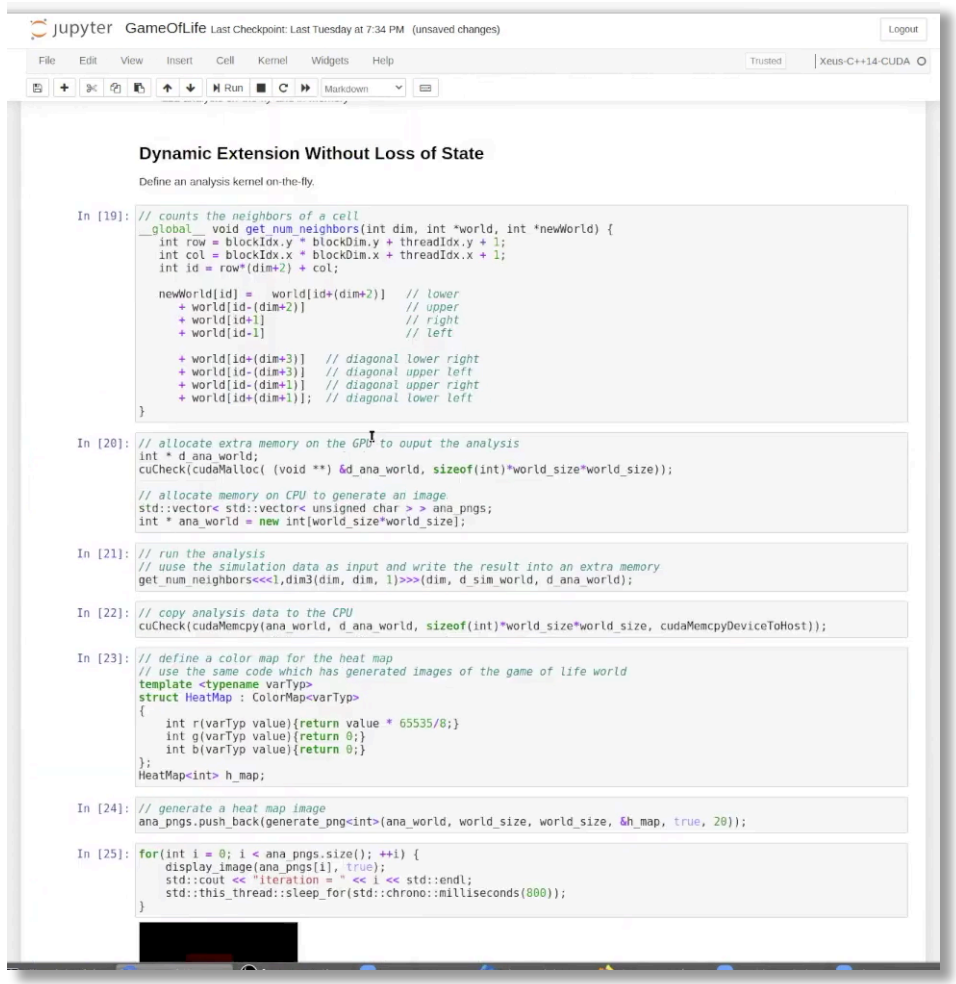
Direct access to documentation

Rich mime type rendering in Jupyter

S. Corlay, Quantstack, *Deep dive into the Xeus-based Cling kernel for Jupyter*, May 2021, compiler-research.org

Interactive CUDA C++





S. Ehrig, HZDR, *Cling's CUDA Backend: Interactive GPU development with CUDA C++*, Mar 2021, compiler-research.org

Automatic Language Bindings With Cling

cppyy: Yet another Python – C++ binder?!

- Yes, but it has its niche: *bindings are runtime*
  - Python is all runtime, so runtime is more natural
  - C++-side runtime-ness is provided by Cling
- Very complete feature-set (not just "C with classes")
- Good performance on CPython; great with PyPy✷

pip: https://pypi.org/project/cppyy/
conda: https://anaconda.org/conda-forge/cppyy
git: https://github.com/wlav/cppyy
docs: https://cppyy.readthedocs.io/en/latest/

For HEP users: *cppyy in ROOT is an old fork. It won't run all the examples here, doesn't work with PyPy, and has worse performance.*

(✷) PyPy support lags CPython

BERKELEY LAB - 2 - U.S. DEPARTMENT OF ENERGY

[1]

sil-cling: Interface with cppyy

- binds with cppyy through the direct inclusion of the latter's C header using dpp

```
1  //cling.dpp
2
3  #include "capi.h"  // cppyy's C header
4
5  // D code ↓
6  import std.string : fromStringz, toStringz;
7
8  string resolveName(string cppItemName)
9  {
10     import core.stdc.stdlib : free;
11     // Calling cppyy_resolve_name ↓
12     char* chars = cppyy_resolve_name(cppItemName.toStringz);
13     string result = chars.fromStringz.idup;
14     free (chars);
15     return result;
16 }
```

```
➜  ~ dub run dpp -- cling.dpp --keep-d-files -c
```

[2]

[1] W. Lavrijsen, LBL, cppyy, Sep 2021, compiler-research.org
[2] A. Militaru, Symmetry Investments, *Calling C++ libraries from a D-written DSL: A cling/cppyy-based approach*, Feb 2021, compiler-research.org

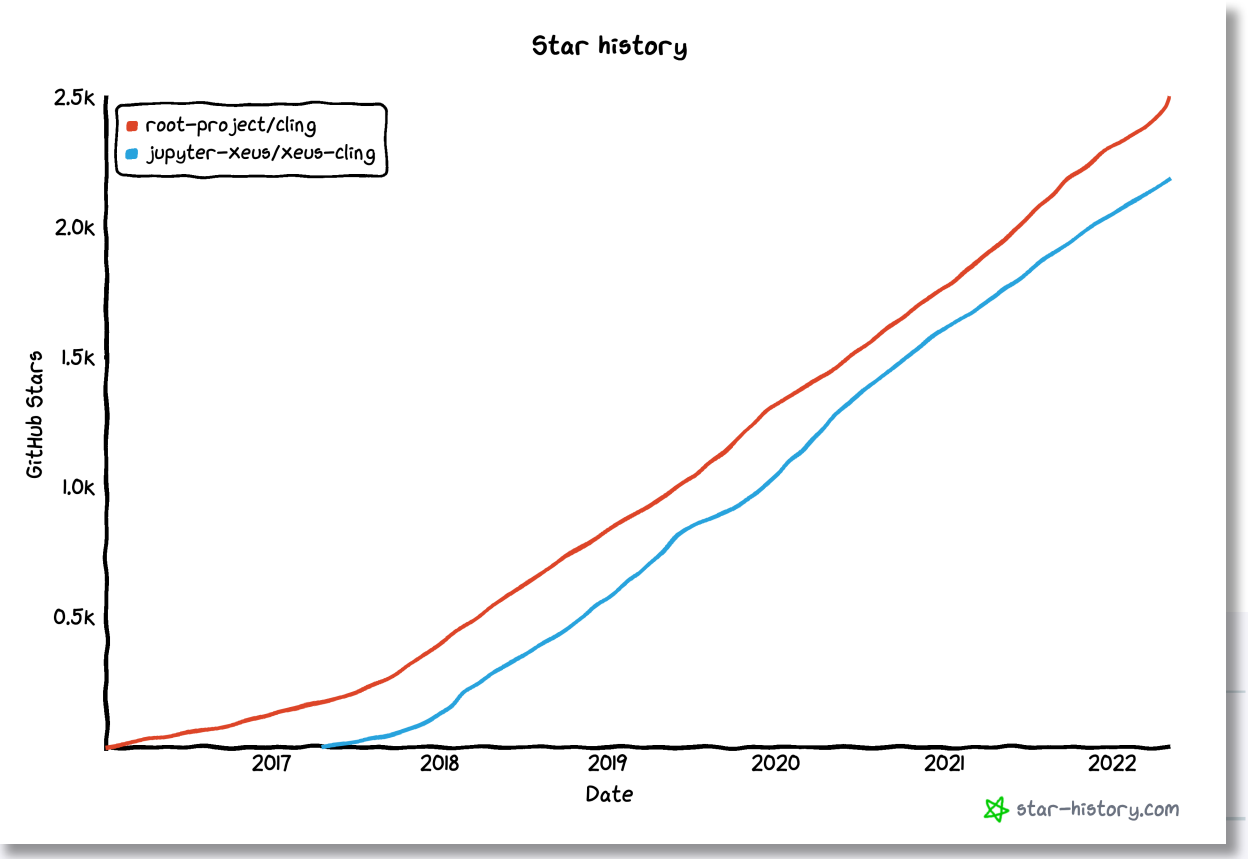
Repository Statistics

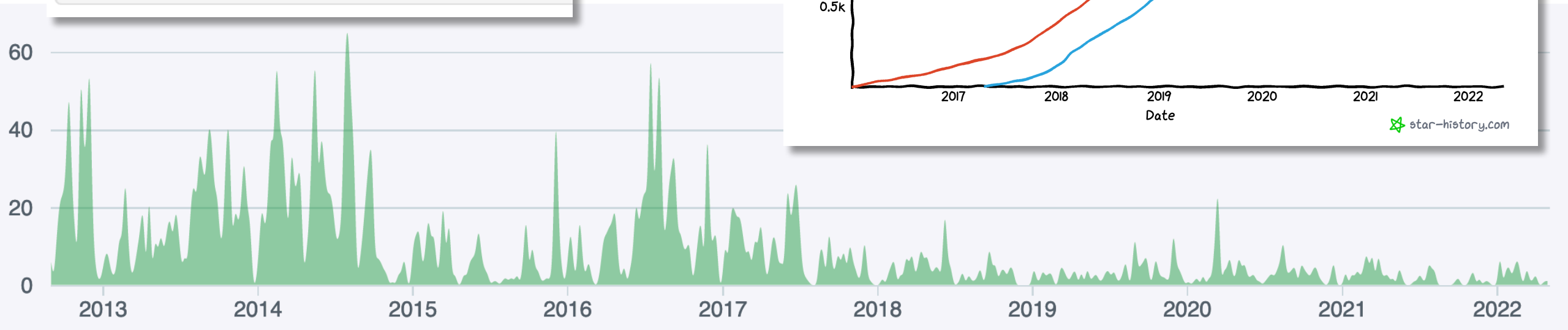
root-project/cling

ROOT-PROJECT/CLING

issue resolution 60 d | open issues 49%

View the details





Moving Cling Closer to LLVM Orbit

- We have demonstrated the power of incremental compilation in ROOT on EB of data
- The growing community around Cling and outside of HEP has shown its relevance to the data science community in general
- Cling's growing community needs better integration in LLVM in terms of release cycles and a stronger connection between the two highly knowledgeable system software engineering communities – the one around LLVM, and the one around data analysis in HEP

The current work is partially supported by National Science Foundation under Grant OAC-1931408

Moving Cling Closer to LLVM Orbit. Clang-Repl

[2018] – We started thinking of dedicated resources to make Cling available to a broader audience

[2019] – We received an NSF award supporting this goal

[2020] – We laid our arguments in a “request for comment” document on the LLVM mailing lists

[2021] – Initial, minimally functional Cling called clang-repl landed in the LLVM repository and was released with LLVM13

[2022] – The LLVM13 upgrade in ROOT uses our work and reduces the codebase of Cling

Thank You!

Selected References

https://blog.llvm.org/posts/2020-11-30-interactive-cpp-with-cling/

https://blog.llvm.org/posts/2020-12-21-interactive-cpp-for-data-science/

https://blog.llvm.org/posts/2021-03-25-cling-beyond-just-interpreting-cpp/

https://Compiler-Research.org