ROOT PoW 2025 – Interpreter

Vassil

- Question: do we want to do another upgrade next year?
- Making things more resilient -> tradeoff speed vs stability
    - There has been some ongoing work on the llvm side
    - How can we make someone else contribute their work for us? (we are a small team)
- We lack test coverage; the more we go towards llvm upstream the more we will/should upstream also tests
- Eventually we should aim to be able to switch llvm version within a week
- JIT-level optimization of virtual calls would be very useful because ROOT interfaces have a lot of virtual functions (e.g. RDF)
    - We have a bunch of PRs that go in that direction

Jonas H.

- two concrete steps:

- reviewing / cleaning up downstream patches

- Review the language extensions we have in cling (e.g. “auto auto”) -> there are probably some more low-hanging fruits that we could get rid of.
    - Can we reduce the cling-specific complexity we have?
    - Philippe: at the same time, we should not reduce the feature set we have arbitrarily

Aaron

- Cling-repl -> lot of interfaces that delegate to pyroot but could be handled in the compiler. E.g. we can offload some lookups to clang
- Lots of benefits in landing interop (e.g. unit tests)
- Adopting interop would use some more standard facilities from clang than using root-meta

Jonas R.

- We're still suffering from memory leaks in pyroot (pyroot doesn’t know about ownership of the value returned from C++)
    - C++ attribute that cling understands that tells python if the user owns the value
- Better support for modern c++ in python: currently a user is penalized for using certain features (e.g. there is auto-casting of raw pointers but not of smart pointers)
    - We should automatically downcast smart ptrs
    - We should have C++20-specific support e.g. transparent conversion between numpy arrays and std::span
- Should we try to reduce patches in respect to upstream cppyy? -> probably not very prioritary since cppyy is not in active development

- Same thing for cling-repl
- HW accelerators support: sycl/cuda
    - Vassil: cuda works pretty well in cling, not working in ROOT. Doable, low hanging fruit to make it work.
    - Dev: we can also run sycl, there is a PR
- JonasR/Vassil: we should not treat cppyy the same way as llvm because their development pace is very different (for llvm we play catch-up, for cppyy we are mainly ahead); if we go for cpp-interop it will be a waste of effort

Dev

- Comment: the existing patches we have don't hurt that much in upgrading, it's mostly a problem of rebasing. Moving from cling to clang-repl would help in this.
- Question: do we still want to move to clang-repl if we want to adopt cpp-interop?
    - Vassil: cpp-interop sits on top of the interpreter
- Vassil: vision: we will have our own fork of llvm (with or without patches) and we have cpp-interop on top of it

Vincenzo

- Mainteinability:
    - Upgrades to future llvm
    - Upgrades to our python packaging systems (conda, pip(?)) -> at least for conda there will always be a need for manual intervention, which can be quite frequent (may be mitigated by nightly builds). E.g. updates to mac-os sdk that change the libc++ version (which we cannot simply pin in conda because we'd force that version on downstream packages). There is a sequence of actions that need to be done before we can update the root conda package, part of which depends on outside organizations (conda feedstock)
    - We will be forced to keep moving to future llvm versions, if anything due to mac os.
    - We need to write down these big efforts on the PoW
- Comment: if we write down something in the PoW we should have "accountability" (a person we can refer to for that specific item)

Philippe

- Make sure the JITted code is debuggable (we see the stack trace etc)
    - This is already supported!
    - Should we make it the default in debug builds?

LOW HANGING FRUITS

- Cuda support in ROOT
- Enabling debug info in JIT
- *Patch review (medium-term)*
    - *Reorganize them to make them self-contained*
    - *See if we can drop any*

    - *Have a clear statement somewhere on why we need a specific patch (to be re-evaluated every upgrade)*
- cpp-interop in the build system
- *Cppyy patches (not backend) -> our patches change the behavior so it is up for debate*

Other things safe to put in the PoW

- Adopt JITcall in call func
- Use interop unit tests to validate the interpreter backend -> ability to configure interop to use either cling or clang-repl and it works the same
- Dynamic library manager
- Sycl support in cling
-