ROOT's Runtime C++ Modules

Vassil Vassilev, Raphael Isemann

Status Quo

- 2016, 1 FTE (Vassil): ROOT compiles and passes roottest when compiled with -Dcxxmodules=On
    - Significant amount of work put into infrastructure, outreach, setting up the right discussion forums to advance the C++ modules feature in clang
- 2017, 1.5 FTE (1.0 Raphael, 0.5 Vassil): ROOT compiles and almost passes (50 failing out of 1650 tests) with -Druntime_cxxmodules=On. Big thanks for the great work done by Raphael!

Future Directions

- Polish the existing support for runtime C++ modules;
    - Optimize the redundant deserializations in both ROOT and LLVM
    - Rethink rootmap-related code in terms of cxxmodules
    - Make clean non-cling-dependent modules (see RE-0003).

Future Directions

- Keep various modules-related nightly builds ensuring correctness and tracking regressions in ROOT but also LLVM;
- Continuous performance monitoring (mainly memory and execution speed);
- Define a path forward for experiments to adopt that feature (preliminary talks with 2 experiments, potentially allocating resources in spring and summer 2018);
- Find potential use-cases of the feature outside the scope of dictionaries, such as package management based on modules infrastructure (see slide “Going Beyond Dictionaries”);
- Provide experiments migration support;
- Work on a long-term community support plan.

Going Beyond Dictionaries

- C++ Modules can be used to give stronger component encapsulation;
- C++ Modules (and their modulemaps) can give enough information for a lean implementation of a distributed package manager (Oksana, Brian and I are working on a proposal)

FAQ

1. **Why it is needed?**
   It is expected to reduce the memory footprint and increase the performance of code using ROOT. We can improve correctness of library autoloading and simplify the plenitude of callback invocations.
2. **Who is expected to use it?**
   Runtime C++ Modules a core feature and the target user group is all users of ROOT, including experiment software stacks.
3. **Why in 2018; can it wait?**
   This is a long waited and long advertised feature. Technically, people can wait one more year (they have been waiting for it for maybe 5 years now). The risk is that waiting more might stamp the project as not being technically feasible to implement and revoke funding.
4. **Timing**
   In 2016 the deliverable was a complete build of ROOT -fmodules flags and successfully passing test suite. It included approx. 100% of Vassil's work time (including a lot of scaffolding being set on the ROOT side but also on the LLVM side). We created a forum where regressions could be tracked and fixed in a timely manner (by cxxmodules engineers). In 2017 we are converging towards enabling modules-aware dictionaries in ROOT and we are working on the last ~50 (out of ~1650) failing tests. Many thanks to Raphael who made excellent progress since Feb 2017. The estimate work is approx 1.5FTE (Vassil's work plan shifted a little because of changed employers). You can read more on the technical part work being conducted by Oct 2016 here. In 2018 the expectancy is 1.5FTE.
5. **Communicating results**
   Using the standard forums root-planning meeting combined with informal chats with the parties involved. Perhaps writing a publication for a HEP conference.