Porting ROOT and Cling to RISC-V

Jonas Hahnfeld[1*]

[1]CERN EP-SFT, Esplanade des Particules 1, 1211 Geneva 23, Switzerland

Abstract

The ROOT data analysis framework is used to analyze exabytes of data in the domain of High Energy Physics (HEP). One critical ingredient is its interactive C++ interpreter Cling, powering many areas of ROOT from the IO layer to interoperability with Python. Outside of HEP, Cling is also used to provide the C++ kernel for Jupyter notebooks. The interpreter is built on top of LLVM and Clang and uses just-in-time compilation (JIT) to emit and execute native machine code. In this contribution, we describe the work required to port ROOT and Cling to the RISC-V architecture. We describe the changes needed to run a first physics analysis and put special emphasis on the particularities for supporting RISC-V's modular ISA design.

Introduction

The ROOT data analysis framework [1] is widely used within the context of High Energy Physics (HEP). It is used to store and analyze exabytes of data that is produced during simulation and data taking in experiments. For example, ROOT is currently used by all experiments at the Large Hadron Collider (LHC), hosted by CERN near Geneva.

One central component of the ROOT framework is Cling, the interactive C++ interpreter built on top of LLVM and Clang [2]. It is critical to a number of areas within the framework: The IO layer queries the interpreter to obtain information about class members and their types. This allows to serialize data into a columnar format that physicists will be able to read back with future versions of ROOT.

Another important part of the framework that relies on Cling is RDataFrame [3, 4]. It provides a declarative interface for physics analyses and needs type information about used data columns that may only be available at runtime after opening input files. Moreover, a significant part of its design is based on just-in-time compilation (JIT) for performant filtering and processing. Finally, Cling is also used to provide interoperability between C++ and Python. For example, ROOT allows users to instantiate templates from Python, as well as calling C++ member functions.

At the time of writing, HEP computing resources are mainly provided by x86, with some experiments exploring ARM and PowerPC. RISC-V is a new instruction set architecture (ISA) and an open standard that does not require paying licensing fees to use or design hardware. Because of that and as first hardware is becoming available, RISC-V is currently gaining traction in the open source world. In that context, we describe porting ROOT and its Cling interpreter to RISC-V to prepare for future experimentation.

*Corresponding author: `jonas.hahnfeld@cern.ch`

LLVM JIT and clang-repl

As Cling is based on LLVM and Clang, we start porting at the level of the JIT in LLVM's `main` development branch. We find that there already exists a JITLink backend for RISC-V, which takes care of resolving relocations in the JIT compiled code. After enabling this JITLink backend by default for RISC-V and further testing, we conclude that this area is already working well for our purposes.

As the next step, we focus on clang-repl which are generic parts of Cling being upstreamed into LLVM. We again find that its basic operations already work on RISC-V, but solve a problem specific to this new architecture: RISC-V is a modular ISA that relies on extensions to add support for floating point instructions. As a consequence, the set of extensions also determines the available registers and the used calling convention. In this case, Linux on RISC-V targets `RV64GC` and assumes support for double-precision floating point, the `D` extension. Clang consequently chooses the Application Binary Interface (ABI) `lp64d` consistent with that. However, the LLVM backend and code generation module complain that this ABI cannot be used with the base ISA. To solve this, we change clang-repl to propagate all assumed "target features" from Clang to the LLVM backend.

ROOT and Cling

For porting ROOT and Cling, we first add support for detecting RISC-V to ROOT's build system and configuration files. In addition, we need a version of LLVM in ROOT that supports JIT compilation for RISC-V. When starting our port, ROOT was still based on LLVM 9 without the JITLink backend for RISC-V described in the previous section. However, there was work underway to upgrade to LLVM 13 which at least contained most of the base work for JIT compilation on RISC-V. For this reason, we base our port on the version of ROOT making use of LLVM 13[1].

Still, it is necessary to backport a number of commits from later versions of LLVM. We also need to apply the changes described for clang-repl to the Cling code base. Furthermore, in Cling we explicitly need to propagate the computed ABI to the machine code generator. Otherwise it will still pass floating point arguments on the stack instead of in registers. This is different from clang-repl which automatically propagates this information based on a different code path in Clang.

Additionally, we find that compiling more complex code makes wider use of compressed instructions. This requires additional relocations for compressed branches and jumps, `R_RISCV_RVC_BRANCH` and `R_RISCV_RVC_JUMP`. We implement support for them locally in LLVM 13 and also contribute the changes upstream.

First Physics Analysis

To test the port to RISC-V, we run some of the RDataFrame tutorials available with the ROOT source code. They document the most important features of RDataFrame and their usage in simplified analyses. Moreover, some tutorials are written directly in C++ while others use Python to call RDataFrame. This means that we are able to test the interaction between the two programming languages, which also relies on Cling. We find that all tested tutorials, `df10*` and higher, work without problems on RISC-V.

The most complex of these tutorials is `df103_NanoAODHiggsAnalysis.py`. It implements a simplified physics analysis in Python, but also uses Cling to just-in-time compile a C++ header file. The interpreted functions are then used to filter events recorded with the CMS detector at the LHC in 2011-2012, now publicly available on the CERN Open Data Portal [5]. After filtering, the analysis computes the invariant mass and plots the histogram in Figure 1. The results show a bump at around 125 GeV, indicating the decay of the Higgs boson.

Conclusions and Future Work

In this contribution, we described our porting of ROOT and Cling to RISC-V. We presented the required changes in the involved software parts, in particular related to ISA extensions. To our knowledge, we also presented the first physics analysis run on this new architecture.

One feature that currently does not work on RISC-V is exception support: If JIT compiled code throws an exception or an exception needs to be propagated through interpreted frames, ROOT will terminate. This can be solved by correctly registering the exception handling information with the unwinder, and we plan to address this in the future.

[1] The upgrade to LLVM 13 was merged into ROOT's `master` branch in December 2022.

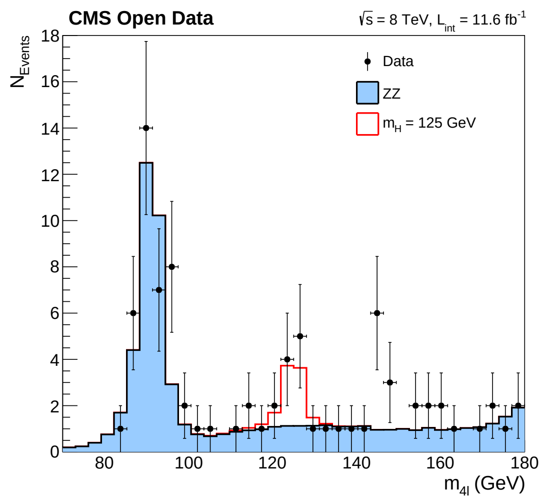

Figure 1: *Output produced by the RDataFrame tutorial* `df103_NanoAODHiggsAnalysis.py`. *It shows the invariant mass of selected events with four leptons, indicating the decay of the Higgs boson at a mass of $m_H = 125$ GeV into two Z bosons.*

Acknowledgment

The author would like to thank RISC-V International and StarFive for providing a VisionFive V1 via the RISC-V Developer Board program.

References

[1] Rene Brun and Fons Rademakers. "ROOT — An Object-Oriented Data Analysis Framework". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 389 (1997), pp. 81–86. DOI: `10.1016/S0168-9002(97)00048-X`.

[2] Vassil Vassilev et al. "Cling – The New Interactive Interpreter for ROOT 6". In: *Journal of Physics: Conference Series* 396.5 (Dec. 2012), p. 052071. DOI: `10.1088/1742-6596/396/5/052071`.

[3] Guilherme Amadio et al. "Novel functional and distributed approaches to data analysis available in ROOT". In: *Journal of Physics: Conference Series* 1085.4 (Sept. 2018), p. 042008. DOI: `10.1088/1742-6596/1085/4/042008`.

[4] Danilo Piparo et al. "RDataFrame: Easy Parallel ROOT Analysis at 100 Threads". In: *EPJ Web Conf.* 214 (2019), p. 06029. DOI: `10.1051/epjconf/201921406029`.

[5] Jomhari, Nur Zulaiha; Geiser, Achim; Bin Anuar, Afiq Aizuddin; (2017). *Higgs-to-four-lepton analysis example using 2011-2012 data.* CERN Open Data Portal. DOI: `10.7483/OPENDATA.CMS.JKB8.RR42`.