An IPCC to modernize the ROOT Math and I/O libraries

Peter Elmer, Vassil Vassilev

PRINCETON
UNIVERSITY

ROOT IPCC objectives

Objectives:

- Code reuse between ROOT and Geant
- Improving performance
- Code Modernisation

The original ROOT IPCC proposal from (nearly) 1 year ago had I/O tasks in Y1 and Math tasks in Y2. In these slides we propose to reorder the tasks to take into account the current situation/opportunities and the expertise of the developer for this project (Vassil, now hired).

Vassil bio

- PhD in Computing
- Works in the ROOT team since 2010
- Authored ROOT’s C++ interpreter, cling
- Part of the ROOT6 development and adoption team
- Contributor to various areas of ROOT and llvm and clang
- Bulgarian, Married

Year 1 Plan

Item	***Deliverable***	***Success Criteria***	***Timeframe***
Plan	Updated work plan for 2017	Approved work plan	Q1
ROOT Math	Integrate VecCore in ROOT. Help with ongoing math vectorisation work.	Speed up the progress of vectorization of ROOT Math	Q2
ROOT Math	Integrate the automatic differentiation prototype, clad, in ROOT.	Adoption in ROOT. Benchmark the performance of using it in fitting (minuit) or training neural networks (TMVA).	Q3
ROOT I/O	Thread-based file merging in ROOT based on a prototype in Geant by Witold Pokorski	Report and a prototype of the general concept.	Q4

ROOT Math

Help adopting VecCore in ROOT:

- Exposes the functionality to ROOT users, including LHC experiments.
- Help with the ongoing work in vectorisation of the Math Libraries.
- Potentially useful in Geant

Major work items:

- Make the system available in ROOT
- Provide runtime information about system’s architecture

ROOT Math

Extend the clang-based automatic differentiator, clad, and adopt it in ROOT:

- Exposes the functionality to ROOT users, including LHC experiments.
- Can be used in ROOT's Math Packages
- Potentially useful in Geant

Work items:

- Make the system available in ROOT
- Benchmark the performance impact in fitting and training neural nets.
- Implement high performant computation of gradients
- Investigate derivative computation on highly parallel architectures (eg. Intel Xeon Phi)

ROOT IO

Extend the thread based-file merging prototype and move it to ROOT.

- CMS is interested in the prototype
- CMS plans to use it and expects significant performance improvements

Work items:

- Trigger discussion and enumerate work plan
- Understand and test the code
- Extend the prototype into a production-ready working system
- Adopt and enhance the prototype
- Benchmark the performance benefits in CMS

Future work

ROOT Math:

- Improve thread safety of RooFit
- Investigate potential uses of clad in RooFit
- More synergy between clad and TMVA

Collaborating project - DIANA/HEP

An NSF-funded project focused on developing tools for the HEP analysis tools ecosystem (of which ROOT is a core element). DIANA/HEP has three broad goals: improving performance, increasing interoperability of HEP tools with the broader scientific software ecosystem and providing tools for collaborative analysis.

For the IPCC, the focus on performance is the relevant part. The IPCC will collaborate with DIANA (and the ROOT team) on I/O and probably (eventually) RooFit modernization.

Team: Princeton, U.Nebraska-Lincoln, U.Cincinnati, NYU

Website: http://diana-hep.org

Related projects - Parallel Kalman Filter Tracking

Charged particle tracking reconstruction is the key pattern recognition algorithm requiring modernization for parallel architectures and the challenges of the HL-LHC.

This is an NSF-funded project which is aiming to modernize these algorithms for use by CMS and others at the HL-LHC.

For the IPCC project, it provides a key testbed and use cases for testing vectorization (e.g. Matriplex, VecGeom)

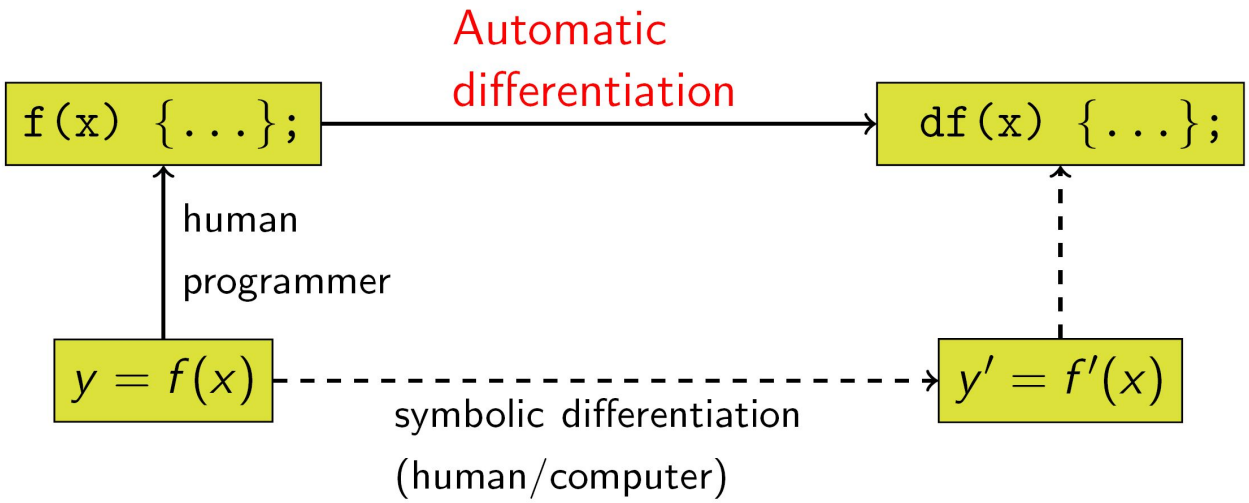
Team: Princeton, UCSD, Cornell

Website: http://trackreco.github.io

Thank you

Backup

Automatic differentiation



https://indico.cern.ch/event/258092/contributions/1588567

Improving buildtime and runtime performance

Reduce ROOT build times

- Move LLVM as an external project
- Build ROOT with C++ modules

Plan:

- Upstream internal LLVM and Clang patches. Expected build time reduction ~50-60%
- Improve the clang-based implementation of C++ Modules. Expected build time reduction ~40%.