New RooFit Developments to Speed up your Analysis

Zef Wolffs (Nikhef), Carsten Burgard (DESY), Jonas Rembser (CERN), Lorenzo Moneta (CERN), Wouter Verkerke (Nikhef), Patrick Bos (Netherlands eScience Center)

ICHEP 2022, 08-07-2022
Physics analyses continuously generate increasingly complex likelihood models to describe their data:
- O(1000) parameters
- O(100) likelihood components
- O(100) datasets

Just to name a few:
- Higgs combination fits
- EFT interpretations

It is certain models will increase in complexity in the foreseeable future with the first LHC run 3 data coming in soon.

RooFit needs to accommodate for these fits:
- From hours fit time down to minutes, i.e. work-day to coffee break.

[1] ATLAS Collaboration. (2020). A combination of measurements of Higgs boson production and decay using up to 139 fb$^{-1}$ of proton–proton collision data at $\sqrt{s} = 13$ TeV collected … (see last slide)
● Physics analyses continuously generate increasingly complex likelihood models to describe their data
  ○ O(1000) parameters
  ○ O(100) likelihood components
  ○ O(100) datasets

● Just to name a few
  ○ Higgs combination fits
  ○ EFT interpretations

● It is certain models will increase in complexity in the foreseeable future with the first LHC run 3 data coming in soon

● RooFit needs to accommodate for these fits
  ○ From hours fit time down to minutes, i.e. work-day to coffee break

[2] ATLAS Collaboration. (2021). Combined measurements of Higgs boson production and decay using up to 139 fb⁻¹ of proton-proton collision data at √s = 13 TeV collected with the … (see last slide)
In which areas does RooFit evolve?

- Vectorization
- Gradient parallelization
- Higher-level interfaces
- GPU Implementation
- Fit precision and correctness
- Pythonizations
- Automatic differentiation
- Testing and benchmarking
- Interoperability
- Performance optimization
- Targeted optimizations for expensive workflows
- User interface and experience
In which areas does RooFit evolve?

- Vectorization
- Gradient parallelization
- GPU Implementation
- Fit precision and correctness
- Automatic differentiation
- Testing and benchmarking
- Targeted optimizations for expensive workflows
- Higher-level interfaces
- Pythonizations
- Interoperability

These are the focus of my talk!
Gradient Parallelization
CPU parallelization - serial MIGRAD

Gradient
N partial derivatives, 2N likelihood evals

line search O(3) evals

Gradient
N partial derivatives, 2N likelihood evals

line search O(3) evals

Gradient
N partial derivatives, 2N likelihood evals

CPU time for partial derivative varies strongly with parameter:

- Some parameters only in subset of likelihood components, so most components do not always need re-evaluation
- Complexity of component likelihood calculation varies
CPU parallelization - parallel MIGRAD

- Parallelize at gradient calculation level
- Dynamic load balancing over workers through random work stealing algorithm
- Designed to have maximum speed impact of complex fits with many parameters
- Line search has limited impact on scaling, but this was investigated further

- Walltime decrease in Higgs combination fit from 2h12m26s → 28m52s
- Serial time expenditure close to parallel time expenditure with single worker
- Fit validated to conform to serial run, all parameters agree within 1% of estimated uncertainty
• Relative time expenditures show that serial components start playing increasingly important role in total walltime when using more workers

• For workspaces with many component likelihoods, parallelizing the linesearch could also prove beneficial

Zef Wolfs, ICHEP 2022
Batched Computation
Batched computation in RooFit refers to the principle of computing batches of events simultaneously, this has the following benefits:

- **GPU parallelization**: Simple operations applied on each of the O(1000)+ CUDA cores on the GPU in parallel
- **Vectorization**: Most modern processors are equipped with instruction sets which apply the same operation simultaneously to multiple pieces of data, or “batches”
Major restructure of RooFit computational back-end to allow for batched computation

- Previous implementation evaluated the computational graph of likelihood components on event-by-event basis

- Newly developed implementation restructures computation graph as sequence of functions to allow for vectorized evaluation of all (relevant) events per computational graph node

- Regardless of vectorization, this new strategy reduces computation times due to improved CPU caching
For unbinned fits with large numbers of events we see a huge speedup using batched computation
  - Vectorization alone effective, but with also multithreading and GPU parallelization (CUDA) we record even larger speedups
  - Speedup grows with the size of the dataset that is being evaluated
    - Especially true for GPU due to relatively large overhead of data transfer to GPU cores
- Plot on the right shows relative time spent in the minimization of the likelihood with Batch mode on vs. off

- Speedup for virtually all tests
  - Even for tests which do not involve large numbers of events, restructuring of the computational graph results in faster minimization

- Speedup appears very dependent on the problem
Conclusion
Conclusion

- Optimization strategies currently under development
  - **Batched computations** (available now, ROOT 6.26)
  - **Gradient parallelization** (available soon, ROOT 6.28)
  - **Combination of the above**, allowing for combined speedup (available soon, ROOT 6.28)
  - **Automatic differentiation** (early stage)
    - Prototyping HistFactory implementation with automatic differentiation using Clad
- Also more tools under development to allow analysers to scrutinize workspace and optimize analysis computationally
- Reduced complex Higgs combination fit time by factor five: goal of reducing day-long fits to coffee break within reach


Parallel CPU fit validation

Hcomb workspace (120 VM steps, Npar=3105, Ncomp=334)

Convergence
99% of the 3105 parameters agree within 0.1% of estimated uncertainty
All parameters agree within 1% of the estimate uncertainty