PROCEEDINGS OF SCIENCE

RooFit Developments to Speed up your Analysis

Zef Wolffs,[a,*] **Patrick Bos,**[b] **Carsten Burgard,**[c] **Emmanouil Michalainas,**[d,e] **Lorenzo Moneta,**[d] **Jonas Rembser**[d] **and Wouter Verkerke**[a]

[a] *Nikhef,*
Science Park 105, Amsterdam, the Netherlands
[b] *Netherlands eScience Center,*
Science Park 402, Amsterdam, the Netherlands
[c] *TU Dortmund,*
August-Schmidt-Straße 1, Dortmund, Germany
[d] *CERN,*
Espl. des Particules 1, Meyrin, Switzerland
[e] *Aristotle University of Thessaloniki,*
Thessaloniki 541 24, Greece

E-mail: zefwolffs@gmail.com

As the field of high energy physics moves to an era of precision measurements its models become ever more complex and so do the challenges for computational frameworks that intend to fit these models to data. This report describes two computational optimizations with which RooFit intends to address this challenge: parallelization and batched computations. For the former, a problem-agnostic parallelization framework was devised with generality in mind such that it could be seamlessly applied at various stages of the existing Minuit2 minimization routine. In the results shown in this report parallelization was applied at the gradient calculation stage. The batched computations approach on the other hand required an overhaul of the current manner in which RooFit prepares its computatational graph for the evaluation of likelihoods. This report includes initial benchmarks of the batched computations strategy run on a CPU with vector instructions. Both strategies show significant performance improvements and the parallelization approach at its current state also proves to be robust enough to consistently fit state of the art physics models to real LHC data. Future developments are targeted towards combining both technologies in RooFit in a production-ready state in a ROOT release in the near future.

41st International Conference on High Energy physics - ICHEP2022
6-13 July, 2022
Bologna, Italy

*Speaker

PoS(ICHEP2022)249

1. Introduction

RooFit [1, 2] has consistently been the main software for researchers fitting large statistical models to the data collected by the four main experiments at the Large Hadron Collider (LHC). As such, it has played a key role in some of the most significant High Energy Physics (HEP) experimental findings in recent history, perhaps most notably the discovery of the Higgs boson in 2012 [3, 4]. This discovery was seen as a significant confirmation of the validity of the standard model, and since this result the field of HEP has not had such an obvious target for future discovery and has thus largely transitioned to precision tests of standard model predictions. However, with these developments come ever-increasingly complex models that are fitted to ever-larger datasets. Whereas the Higgs discovery was initially done by testing models where the hypothesized Higgs boson decays through specific interactions, current hypotheses aim to test simultaneously all (beyond) standard model Higgs production and decay modes in a single large fit in an aim to find deviations from predictions in such a global description [5]. For any statistical software to become the tool of choice in this era of precision measurements it is crucial that it is able to perform these fits in both a robust and efficient manner. The research presented in this report provides an overview of two major recent developments in RooFit that intend to deliver on the efficiency requirement by providing order of magnitude speedups to the aforementioned fits while not compromising on their robustness. The following chapter provides a short description of these efficiency-improving methods.

2. Methods

2.1 Parallelization

The most common statistical formalism for testing models given data in the field of HEP is that of likelihood estimation. In this context, the likelihood function $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x})$ describes the joint probability between observed data $\boldsymbol{x}$ as a function of the parameters $\boldsymbol{\theta}$ of the chosen statistical model. Of physics interest are those model parameters that describe the data best, which by definition are the parameters that satisfy the maximum value of the likelihood function. To find this point in parameter space, in general the negative logarithm of the likelihood (NLL) is minimized.

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}) \Rightarrow -\log \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}) \tag{1}$$

As such, fitting a physics model to data can be generalized to an ordinary minimization problem where $-\log \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x})$ is minimized as a function of $\boldsymbol{\theta}$. RooFit uses an iterative algorithm, Migrad, for such minimization problems in ROOT's Minuit2 package [6]. While it is significantly more sophisiticated, for the purpose of this report it suffices to see the Migrad algorithm as one that iteratively steps to a minimum by using gradient information to calculate a step direction, and likelihood evaluations to calculate a step size in the found direction.

Previous benchmarks [7] showed that besides serial improvements including vectorization—which is the subject of the next section—Minuit2 suffered from two major performance bottlenecks that could be solved by parallelization. Ordered by expected speedup gain these were: the calculation of gradients to calculate the step direction and the evaluation of the likelihood in the line search step.

The first two points were tackled in the present research. A general parallelization framework `RooFit::MultiProcess` was devised to centralize the logic of these and any potential future parallelization efforts. This framework is completely agnostic as to the minimization problem, leaving the implementation detail of how to split the workload into tasks that it can execute in parallel to its caller. For gradient calculation the workload is split up by partial derivatives, whereas for the line search calculation the workload is split up by likelihood component evaluations.

At the start of a new workload the parallel framework receives a set of tasks (partial derivatives in the case of a gradient calculation, likelihood evaluations in the case of a line search calculation) and assigns them to a single queue process to do the bookkeeping of these tasks. As soon as this queue process has work available, the other processes—the workers—start to repeatedly steal individual tasks from the queue, executing them, and sending back the results to the calling code after each finished task until the queue is empty. Note that work stealing algorithms, of which the algorithm described here is a simplified variant, have the advantage that they scale close to optimal in general cases [8, 9].

2.2 Batched Computations

As stated previously, another major performance bottleneck in evaluating the NLL could be overcome by means of vectorization, i.e. the application of simple logic operations to a vector of datapoints simultaneously, rather than one by one. Modern day processors often are equipped with logic chips tailored to do these vectorized operations, and compilers capitalize on this by transforming e.g. simple for-loops to vectorized operations. To see where this could be of advantage in this context, we have to work out our understanding of the likelihood in this context a bit further. In HEP, different components of the total model—likelihood components—are often fit to different subsets of a given dataset simultaneously. Figure 1 includes a simplified example of such a model comprised of an NLL with two components, each with two relevant events.

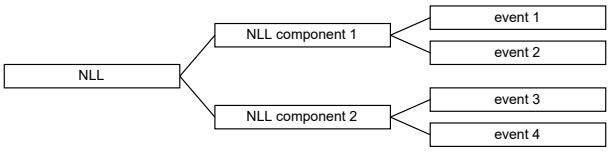


Figure 1: Simplified likelihood model with two likelihood components and four events.

Previously, RooFit would iterate over events in its outer loop, evaluating all relevant components for every event before moving on to the next. In the model illustrated in figure 1 event 1 would propagate through the computational graph first, requiring its evaluation in NLL component 1, then event 2 would walk the same path and so forth. The batched computations mode however instead visits model components in its outer loop, evaluating them one by one for the complete relevant subset of events. In figure 1 this would entail evaluating NLL component 1 for all relevant events, in this case event 1 and event 2, and then doing the same for NLL component 2. This means that model components can be evaluated in a vectorized way, which results in speed improvements by a factor of up to ten when using a single CPU thread. This speedup is not only due to vectorization, but also because of the higher CPU caching efficiency and the fewer virtual funciton calls in the code path. Furthermore, the batch mode can also offload the computation to a GPU if the model

components allow for it. If all components support the evaluation on the GPU, the batch mode is expected to deliver up to fifty times speedup for particular cases.

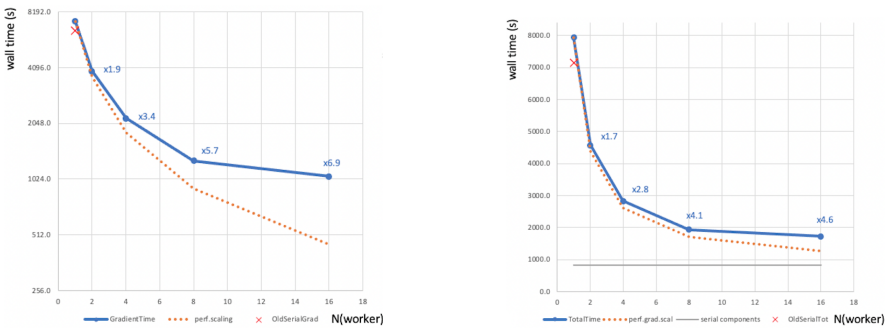
In the case where only a subset of model components are supported on the GPU, copying intermediate results form host to device memory becomes necessary. In order to support this case—and also to be more efficient in general—the computational backend that RooFit uses for likelihood evaluations had to be modified. Previously, each node in the computation graph would query its serving nodes for results recursively, but there was no single entity that could analyze the computation graph and decide where each node gets evaluated to make maximum use of the available resources. This is now done by a new class called the `RooFitDriver`, which compiles the computation graph to a list of computation functions that are called sequentially, with the relevant memory copies in between.

3. Results

3.1 Gradient Parallelization

Gradient parallelization was tested on a workspace with a realistic level of complexity, comparable to those used for recent Higgs combination results. This workspace included 334 likelihood components and 3105 parameters.

The plots in figure 2 show scaling behaviour as a function of the number of workers. Figure 2(a) includes only the time taken within the gradient calculation, whereas 2(b) displays the time taken for the entire minimization. Note that actually two more cores than the reported number of workers were used for each run, where the two extra cores were used to run the queue and caller processes. In terms of total time expenditure the execution time was reduced from 2 hours and 12 minutes down to 29 minutes for 16 workers.



(a) Gradient scaling, on a log scale.

(b) Total scaling, on a linear scale

Figure 2: Scaling of the gradient calculation time and total wall time in seconds versus the number of workers.

Further scrutinization of the parallel code was done by analyzing relative time expenditures as a function of the number of workers, the results of which are included in figure 3. Note that whereas the line search only forms a small part of the total walltime in the cases with smaller numbers of workers, it starts to become more relevant as the gradient walltime decreases with more workers. This confirms the aforementioned earlier benchmarks, which suggested that parallelizing

the gradient was a reasonable priority. However, this also shows that there is still a significant added benefit to parallelizing the line search step for highly parallel runs.

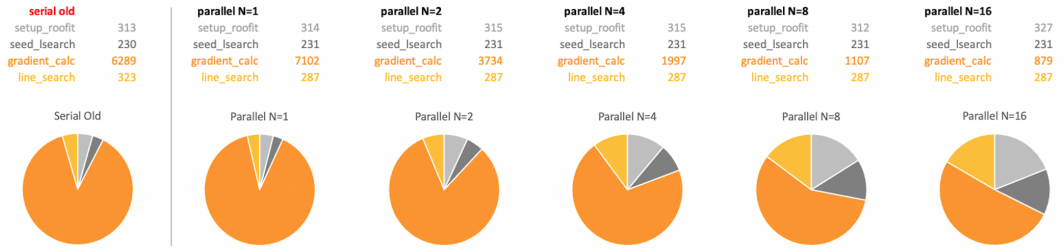


Figure 3: Relative time expenditures as a function of the number of workers. Note that the non-parallelizable parts of the pie charts are grayed out.

An analysis of the results and minimizer convergence in the parallel and serial runs confirmed that 99% of the 3105 likelihood parameters agree within 0.1% of their estimated uncertainty, and that all parameters agree within 1% of the estimated uncertainty. These differences are significantly smaller than the expected numerical precision of the central values as estimated by Migrad with the default tolerance setting of EDM=10^{-3}.

3.2 Batched Computations

The batched computations approach was tested on the set of all likelihood fits in the RooFit tutorials. In these fits, an average speedup of approximately a factor 2.5 was observed (see figure 4). This is lower than the aforementioned expected factor ten, because for some components the batch mode falls back to the scalar implementation. As of now, support for full-sized Higgs combination fits remains work in progress.

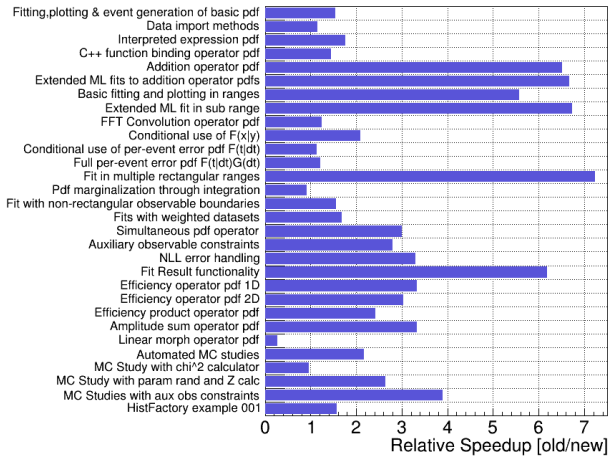


Figure 4: Walltime speedup achieved with the batched computations approach.

4. Discussion & Conclusion

Both implemented efficiency improvements significantly reduced walltimes in all of the test-cases. In its current state, the parallel framework was able to provide a nearly five times speedup

robustly for at least three state of the art HEP test cases, including a Higgs combination fit of which the results were shown in this report. The batched computations approach shows promising results, and development is currently ongoing to reduce scalar computations and further improve its performance.

Future work is primarily oriented towards getting the batched computations approach to work for realistic HEP workspaces. After this, since both implementations are completely independent and optimize the minimization procedure at a different level, their combination is a feasible target, leading to even larger combined speedups. Further optimizations to the parallelization approach on the other hand could include better load balancing and scheduling. Initial simulations on different scheduling approaches show that at least another 10% in further speedup gains should be achievable even without increasing the number of workers.

The presented parallelization developments and batched computations approach will become available in a production-ready state for end users in a ROOT release in the near future, with the latter already available as an experimental feature.

References

[1] Wouter Verkerke and David Kirkby. The roofit toolkit for data modeling, 2003.

[2] Lorenzo Moneta, Kyle Cranmer, Gregory Schott, and Wouter VERKERKE. The RooStats project. *PoS*, ACAT2010:057, 2011.

[3] ATLAS Collaboration. Observation of a new particle in the search for the standard model higgs boson with the ATLAS detector at the LHC. *Physics Letters B*, 716(1):1–29, sep 2012.

[4] CMS Collaboration. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Physics Letters B*, 716(1):30–61, sep 2012.

[5] Combined measurements of Higgs boson production and decay using up to 139 fb^{-1} of proton-proton collision data at $\sqrt{s}$ = 13 TeV collected with the ATLAS experiment. Technical report, CERN, Geneva, 2021.

[6] M. Hatlo, F. James, P. Mato, L. Moneta, M. Winkler, and A. Zsenei. Developments of mathematical software libraries for the LHC experiments. *IEEE Trans. Nucl. Sci.*, 52:2818–2822, 2005.

[7] E. G. Patrick Bos, Carsten D. Burgard, Vincent A. Croft, Inti Pelupessy, Jisk J. Attema, and Wouter Verkerke. Faster RooFitting: Automated parallel calculation of collaborative statistical models. *J. Phys. Conf. Ser.*, 1525(1):012041, 2020.

[8] R.D. Blumofe and C.E. Leiserson. Scheduling multithreaded computations by work stealing. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 356–368, 1994.

[9] Rob V. Van Nieuwpoort, Gosia Wrzesińska, Ceriel J. H. Jacobs, and Henri E. Bal. Satin: A high-level and efficient grid programming model. *ACM Trans. Program. Lang. Syst.*, 32(3), mar 2010.