Efficient Parallelization of RooFit Computations for Accelerated Higgs Combination Fits

Zef Wolffs[1,2,*], *Patrick* Bos[1,3,**], *Lydia* Brenner[1], *Wouter* Verkerke[1,2], and *Ivo* van Vulpen[1,2]

[1]ATLAS Group, Nikhef, Amsterdam, The Netherlands
[2]Institute of Physics, University of Amsterdam, Amsterdam, The Netherlands
[3]Netherlands eScience Center, Amsterdam, The Netherlands

Abstract. In the context of High Energy Physics (HEP) analyses the advent of large-scale combination fits forms an increasing computational challenge for the underlying software frameworks on which these fits rely. RooFit, being the central tool for HEP statistical model creation and fitting, intends to address this challenge through an efficient and versatile parallelisation framework on top of which two parallel implementations were developed in the present research. The first implementation, the parallelisation of the gradient, shows good scaling behaviour and is sufficiently robust to consistently minimize real large-scale fits. The latter, the parallelisation of the line search, is still work in progress for some specific likelihood components but shows promising results in realistic testcases. Enabling just gradient parallelisation speeds up the full fit of a recently published Higgs combination from the ATLAS experiment by a factor of 4.6 with sixteen workers. As the improvements presented in this research are currently publicly available in ROOT 6.28, we invite users to enable at least gradient parallelisation for robust accelerated fitting with RooFit.

1 Introduction

Twenty years after its inception RooFit [1, 2] has remained the central tool for High Energy Physics (HEP) model creation and fitting. During this period it has contributed significantly to many important physics results, not least in the discovery of the Higgs boson in 2012 [3, 4]. Recently though, the physics aims of the large LHC experiments have started to more prominently include global searches in which many more parameters are constrained simultaneously in larger areas of phase space. For example, in a recent ATLAS Higgs combination analysis [5], thousands of free-floating parameters were fit to data simultaneously in hundreds of likelihood components. Such fits are, perhaps unsurprisingly, computationally rather demanding and as a result very time-consuming. Scientists having to wait for their fits to finish for hours, which is often the case for recent combination analyses, is suboptimal for the analysis workflow and might as a result even lead to decreased analysis performance as there is less time for a trial-and-error approach with prompt evaluation of potential improvements.

In the present research, we intend to address this computational challenge by means of a robust and versatile parallelisation framework `RooFit::MultiProcess` which lives under

*e-mail: zefwolffs@gmail.com
**e-mail: p.bos@esciencecenter.nl

the hood of RooFit. This framework was built with generality in mind and as such serves as the necessary back-end to any future parallelisation efforts within RooFit. Two such efforts were implemented and are the subject of this paper: the parallelisation of the gradient and the line search.

The following section (2) briefly introduces the problem and the necessary background on statistical modeling and minimization. Section 3 subsequently introduces the line search and gradient parallelisation strategies. In section 4 the results of both optimizations are presented. And finally section 5 concludes the paper with a brief summary and potential avenues for future development.

2 Background

In the context of experimental particle physics, hypotheses about the processes that occur within particle colliders are expressed in terms of statistical models called likelihood functions. In mathematical terms these likelihood functions, $\mathcal{L}(\theta; \mathbf{x})$, quantify the probability of the observed data $\mathbf{x}$ to occur under the hypothesis encoded by the model and its accompanying parameters θ. As we want to find those model parameters that actually describe the data best, the goal in general is to maximize $\mathcal{L}(\theta; \mathbf{x})$ with respect to θ. Furthermore, since the logarithm has better properties for the purpose of optimization, and since in general most optimization algorithms actually minimize rather than maximize some function, we take the negative logarithm of $\mathcal{L}(\theta; \mathbf{x})$ which we minimize instead.

Since we now expressed the problem of matching hypotheses to data as an ordinary minimization task we have the freedom to choose a minimizer accordingly. RooFit uses the MIGRAD minimization algortihm included in Minuit2 [6, 7], which similarly to RooFit is also included in the ROOT collection of statistical softwares. MIGRAD iteratively steps towards a minimum, determining at every iteration a step direction by means of a gradient calculation, and a step length with a line search calculation. While the full algorithm comprises many more details, the function evaluations in the line search and gradient calculation components are most relevant in this context since previous work [8] has shown that these are the most time-consuming, making them obvious targets for optimization. Conveniently, both are well-parallelisable, the approach of which is described in the next section.

3 Methods

Computationally, the line search step is dominated by a few likelihood function evaluations. Two ways of parallelising these function evaluations were implemented in this work. In the first, the input dataset is split and the likelihood is evaluated for each of the resulting smaller datasets independently, after which the results are summed together. This is called event-based splitting since each parallel task essentially comprises the evaluation of a subset of the full number of events. The second implemented parallelisation strategy is to split the likelihood into independent sets of components, and then to evaluate these sets of components in parallel, again summing the results afterwards. This strategy is called component-based splitting.

The gradient parallelisation on the other hand was implemented by splitting the calculation of the gradient into sets of constituent partial derivative calculations. Being independent calculations, these sets of partial derivatives can then be evaluated fully in parallel.

`RooFit::MultiProcess` serves as the foundational layer for these and any potential future parallelisation efforts in RooFit. It includes three distinct processes: master, worker, and queue.

- **The master** process makes the initial call to `RooFit::MultiProcess`. It lives for the entire duration of the minimization. It is responsible for creating a set of parallel tasks at the beginning of a parallel evaluation, those tasks being for example the sets of partial derivatives for parallel gradient calculation. Master forks the queue and any necessary worker processes at the start of a parallel evaluation.
- **The queue** is responsible for keeping track of and—on worker request—distributing to workers the set of tasks which the master process has generated at the start of the parallel evaluation. It is fully agnostic to the minimization problem that is being executed, and is killed by the master process after no more parallelization is required. Usually this is directly after a likelihood fit.
- **The worker** repeatedly requests and subsequently executes tasks that it receives from the queue process. All workers are killed, along with the queue process, after no more parallelisation is required.

A simplified schematic of the parallel workflow is included in figure 1. Note that for the sake of illustration the calling code and the master process are indicated as two separate entities, but that actually the calling code is the master process in the context of `RooFit::MultiProcess`.

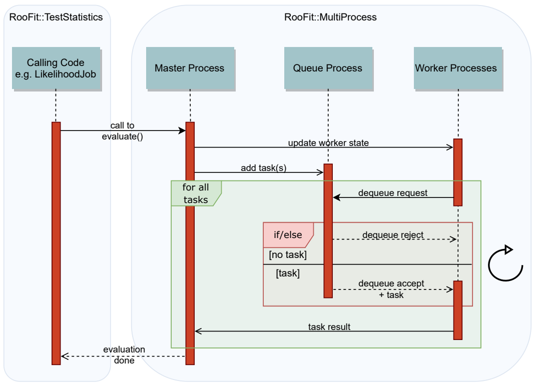

Figure 1. Simplified schematic of the main workflow of `RooFit::MultiProcess`. An explicit distinction is made between the calling code and the multiprocessing framework, indicating at what stage the latter takes over the workflow and where it subsequently hands it back to the former.

Since it was found that randomly ordered parallel tasks could form a major bottleneck for parallel programs with varying task durations the queue was further optimized to handle custom ordering. This allows the user to specify an order for the tasks to be executed. An example of a gradient calculation with tasks ordered by duration can be seen in figure 2. Ordering by duration can significantly improve evaluation times. If, for example, in the gradient calculation in figure 2 the longest task were evaluated last, the whole parallel gradient calculation would have taken significantly longer with most workers idling while a single worker is executing the final task.

4 Results

This section includes results of walltimes for both the gradient and line search parallelisations as a function of the number of workers. Note that for all of the presented results, the likelihood

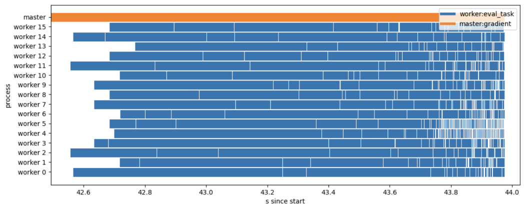

Figure 2. Chart indicating time spent by parallel processes in one full gradient calculation with ordered tasks. The blue bars indicate the duration walltimes of the individual partial derivative calculations executed by the workers. The master process is most of the time waiting for the workers to finish these calculations, which is indicated by the long orange bar. In the blank sections where the workers are not yet executing tasks but the gradient calculation has already begun, the workers are synchronizing their states following the preceeding line search, which can vary in duration depending on how desynchronized their states were at the beginning of the gradient calculation. Note in particular how due to the ordering the larger tasks are executed at the beginning, while the smaller tasks are placed more towards the end.

model and dataset used were those of a recent Higgs combination result from the ATLAS experiment [5]. This likelihood model contains 3105 free-floating parameters and the dataset is the LHC's full run 2 140 fb^{-1} dataset as collected by the ATLAS experiment.

The results for the scaling of the gradient are included in figure 3. Good scaling behaviour is observed for small numbers of workers. When increasing the number of workers a divergence from ideal scaling can be observed as parallel inefficiencies such as communication overhead start to play a larger role. Note that for both figures 3 and 4 ideal scaling is calculated as the walltime of the parallel run with a single worker divided by the number of workers. The serial walltime is also indicated, which in figure 3 is slightly better than the parallel walltime with a single worker, as the latter includes the overhead of communication between the master, worker, and queue processes.

The results for the line search scaling are included in figure 4. Note that for these results, the $H \to \gamma\gamma$ likelihood components were removed from the full likelihood model since they include custom p.d.f. (probability density function) types built on top of the RooFit core library, for which no parallel implementation was developed yet. In the scaling of the line search overall a similar behaviour to the parallel gradient calculation is observed, where the measured walltime diverges from the ideal walltime as the number of workers increases. Scaling performs slightly worse compared to gradient parallelisation, likely because the line search calculation duration is already relatively short, and thus any constant overheads induced by parallelisation have a larger relative impact. Perhaps surprisingly, the parallel implementation here is faster even for a single worker compared to the serial implementation. This is likely since caching was also optimized in the parallel implementation.

For the full Higgs combination analysis including all serial parts the speedup with gradient parallelisation and sixteen workers was measured to be 4.6, reducing the total walltime from two hours and twelve minutes down to twenty nine minutes. Furthermore, an analysis of the results showed that 99% of parameters converged to within 0.1% of their estimated uncertainty with respect to the serial run. All of the 3105 parameters agreed to within 1%. As such, all variations were fully within the numerical precision around the central values estimated by MIGRAD with default tolerance setting of Expected Distance from the Minimum (EDM) at 10^{-3}. Note that a full run with both line search and gradient parallelisation was not done for this research as it would not be representative of a realistic case without the $H \to \gamma\gamma$

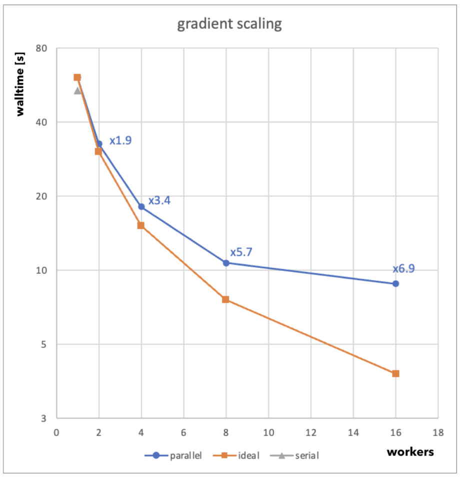

Figure 3. Parallel scaling of the gradient calculation. Data was collected at the annotated markers, with linear extrapolations inbetween. Note that for two and four workers scaling is close to ideal, but that the difference between ideal and observed walltime becomes larger with eight and especially sixteen workers.

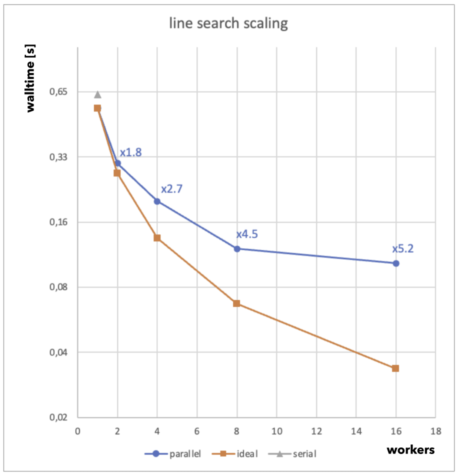

Figure 4. Parallel scaling of the line search calculation. Data was collected at the annotated markers, with linear extrapolations inbetween. Similarly to gradient parallelisation, observed walltimes diverge from ideal as the number of workers increases.

components. However, assuming the speedup observed in the line search parallelisation can be achieved alongside gradient parallelisation for the full Higgs combination fit we can reasonably expect a combined speedup of 5.3 with sixteen workers, reducing the total walltime down to twenty five minutes.

5 Conclusions

This article presented the development and results of a foundational multiprocessing layer which was built for RooFit, on top of which two parallel implementations were developed for the line search and gradient calculations. Both implementations showed significant speedup, with close to ideal scaling for small numbers of workers. The full Higgs combination fit runtime was reduced from over two hours to under half an hour using only gradient parallelisation, as line search parallelisation on this fit is still work in progress. All of the work included in this research is readily available in ROOT 6.28, and users are invited to enable it in their likelihood fits by supplying the '`Parallelize`' keyword argument to `RooAbsPdf::fitTo()`.

A valuable contribution in future research would be the parallelisation of the Hessian calculation in RooFit, which could be built directly on top of the `RooFit::MultiProcess` framework developed in this research. Furthermore, the ordering of parallel tasks by duration showed significant speedup, but is currently only done manually and therefore requires user input. In principle, after only a single parallel evaluation the optimal task ordering for that type of computation can be inferred from walltimes collected on the fly. This automated way of ideally ordering parallel tasks could provide another valuable future development.

References

[1] W. Verkerke, D. Kirkby, *The RooFit toolkit for data modeling*, in *Statistical Problems in Particle Physics, Astrophysics and Cosmology* (World Scientific, 2006), pp. 186–189

[2] L. Moneta, K. Cranmer, G. Schott, W. Verkerke, *The RooStats project*, in *Proceedings of 13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research* (PoS(ACAT2010), 2011), Vol. 093, p. 057

[3] ATLAS Collaboration, Physics Letters B **716**, 1 (2012)

[4] CMS Collaboration, Physics Letters B **716**, 30 (2012)

[5] ATLAS Collaboration, *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*, Geneva (2021), `https://cds.cern.ch/record/2789544`

[6] M. Hatlo, F. James, P. Mato, L. Moneta, M. Winkler, A. Zsenei, IEEE Trans. Nucl. Sci. **52**, 2818 (2005)

[7] F. James, M. Roos, Computer Physics Communications **10**, 343 (1975)

[8] E.P. Bos, C.D. Burgard, V.A. Croft, I. Pelupessy, J.J. Attema, W. Verkerke, *Faster RooFitting: Automated parallel calculation of collaborative statistical models*, in *Journal of Physics: Conference Series* (IOP Publishing, 2020), Vol. 1525, p. 012041