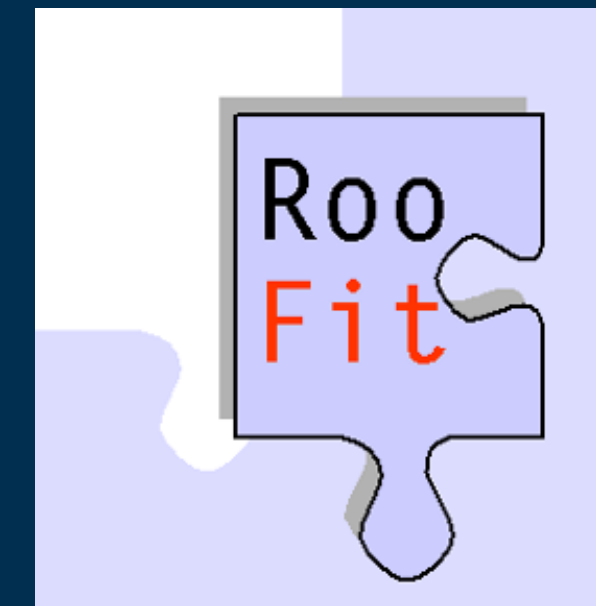


Build-a-Fit: RooFit Parallelisation and Benchmarking Tools

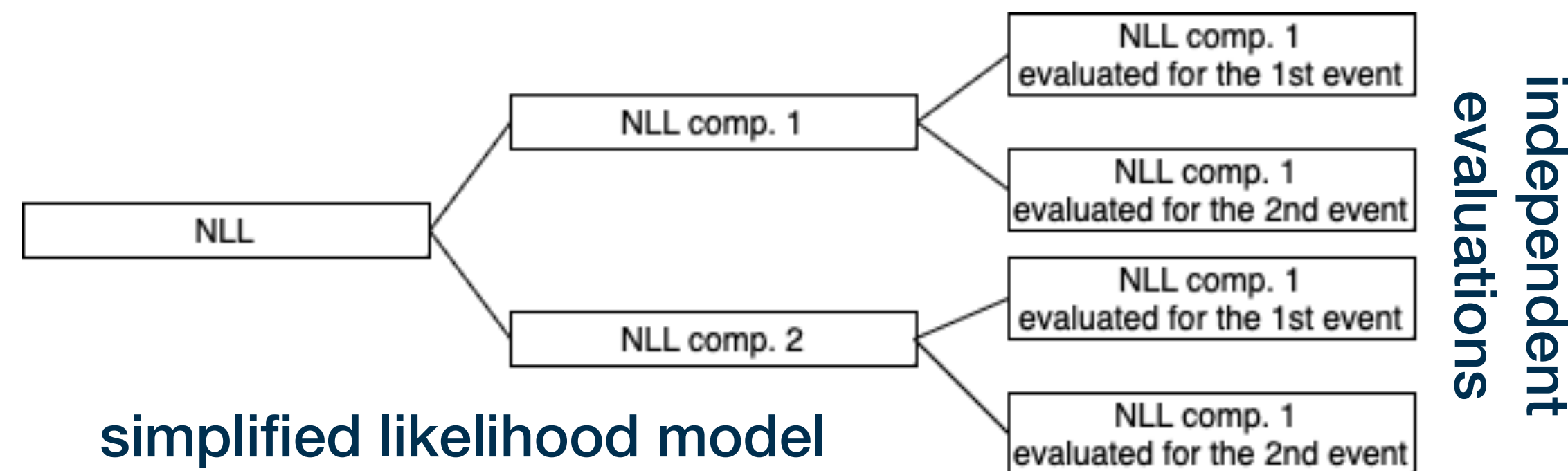


Zef Wolffs (Nikhef, speaker), Patrick Bos (Netherlands eScience center),
Lydia Brenner (Nikhef), Wouter Verkerke (Nikhef), Ivo van Vulpen (Nikhef)

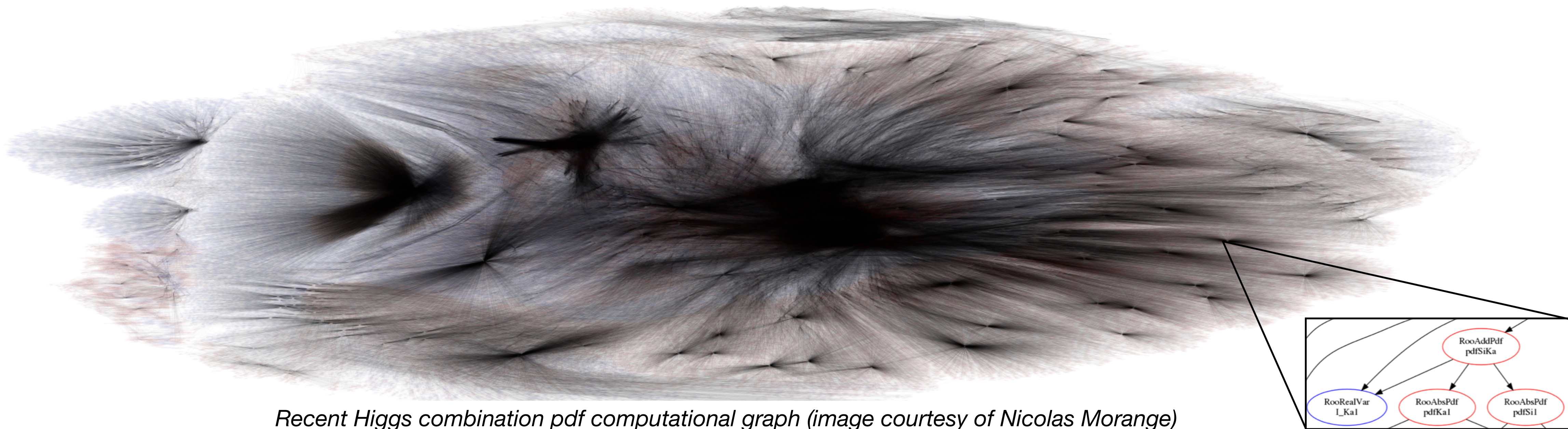
- In high energy physics, hypothesis testing is done by fitting likelihood models to datasets
- **In principle**, parallelising this problem is not hard, remember the likelihood model

$$-\log L(\theta | \mathbf{x}) = -\log \prod_{i=0}^N p(\mathbf{x}_i | \theta) = -\sum_{i=0}^N \log(p(\mathbf{x}_i | \theta)) = \underbrace{-\log(p(\mathbf{x}_1 | \theta))}_{\text{parallel task 1}} - \underbrace{\log(p(\mathbf{x}_2 | \theta))}_{\text{parallel task 2}} - \dots$$

- The evaluation of each event can be calculated fully independently and thus in parallel
- Even more so, likelihood models in high energy physics are generally also constructed from independent components which could also be evaluated in parallel

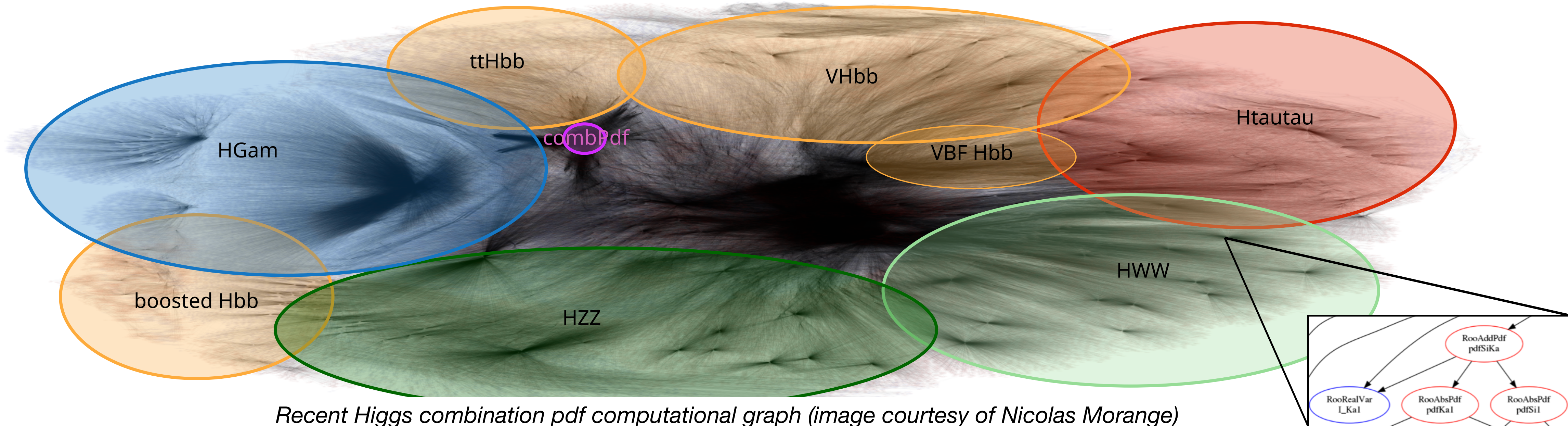


- In practice though, models quickly grow quite convoluted, Higgs combination fits for example incorporate hundreds of smaller likelihood models with varying structures and data
 - This makes it hard to find any general parallelisation strategy with optimal load balancing



- The above likelihood models are those with the longest fit durations, currently taking hours
- The challenge at hand: **Developing a multiprocessing strategy to significantly speed up these complex fits while not compromising on robustness**

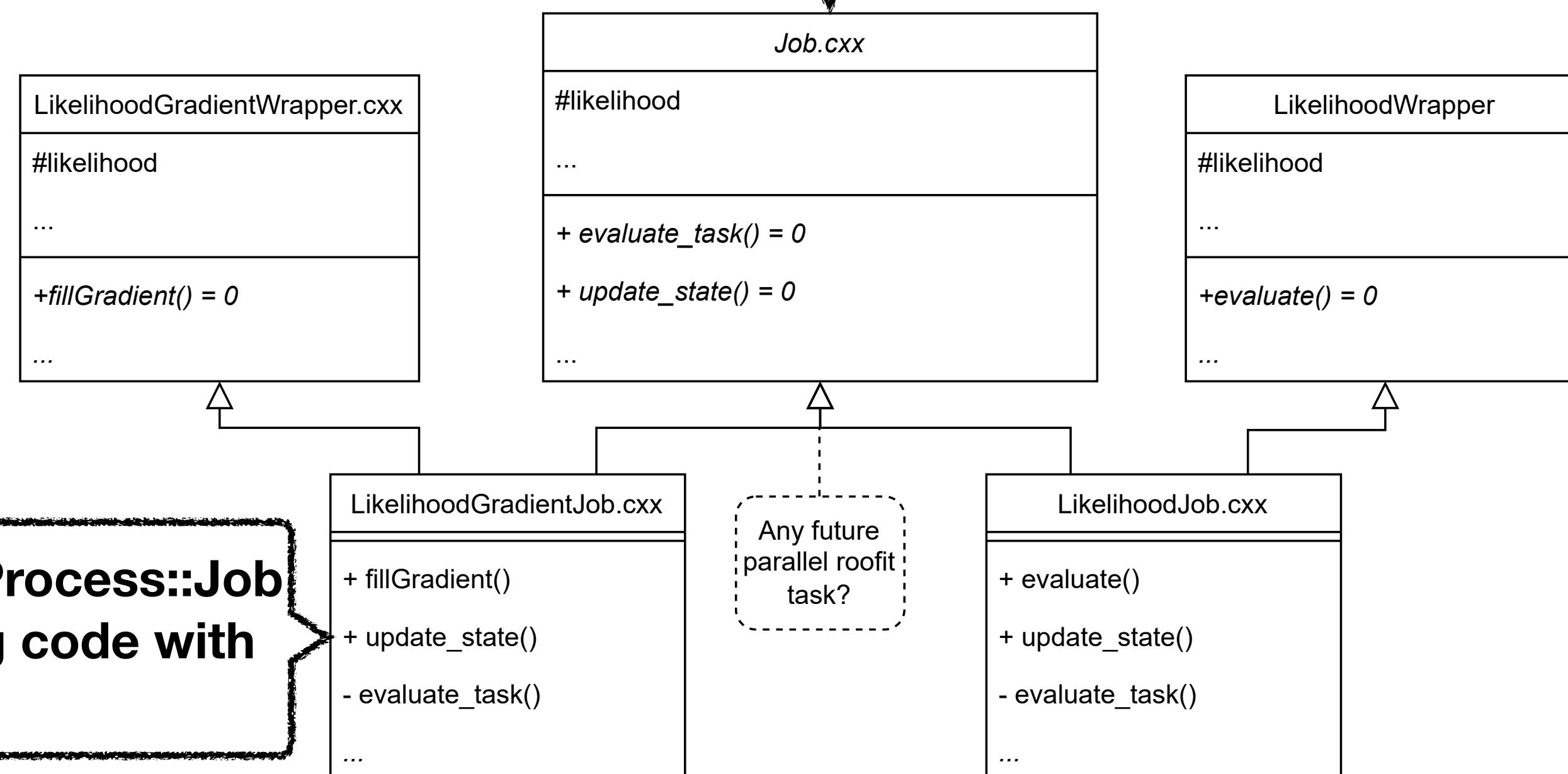
- In practice though, models quickly grow quite convoluted, Higgs combination fits for example incorporate hundreds of smaller likelihood models with varying structures and data
 - This makes it hard to find any general parallelisation strategy with optimal load balancing



- The above likelihood models are those with the longest fit durations, currently taking hours
- The challenge at hand: **Developing a multiprocessing strategy to significantly speed up these complex fits while not compromising on robustness**

- A general parallel framework `RooFit::MultiProcess` was written to serve as a foundation for any RooFit parallelisation efforts
 - Uses ZMQ for interprocess communication
 - Interfaces with rest of RooFit through `RooFit::MultiProcess::Job`

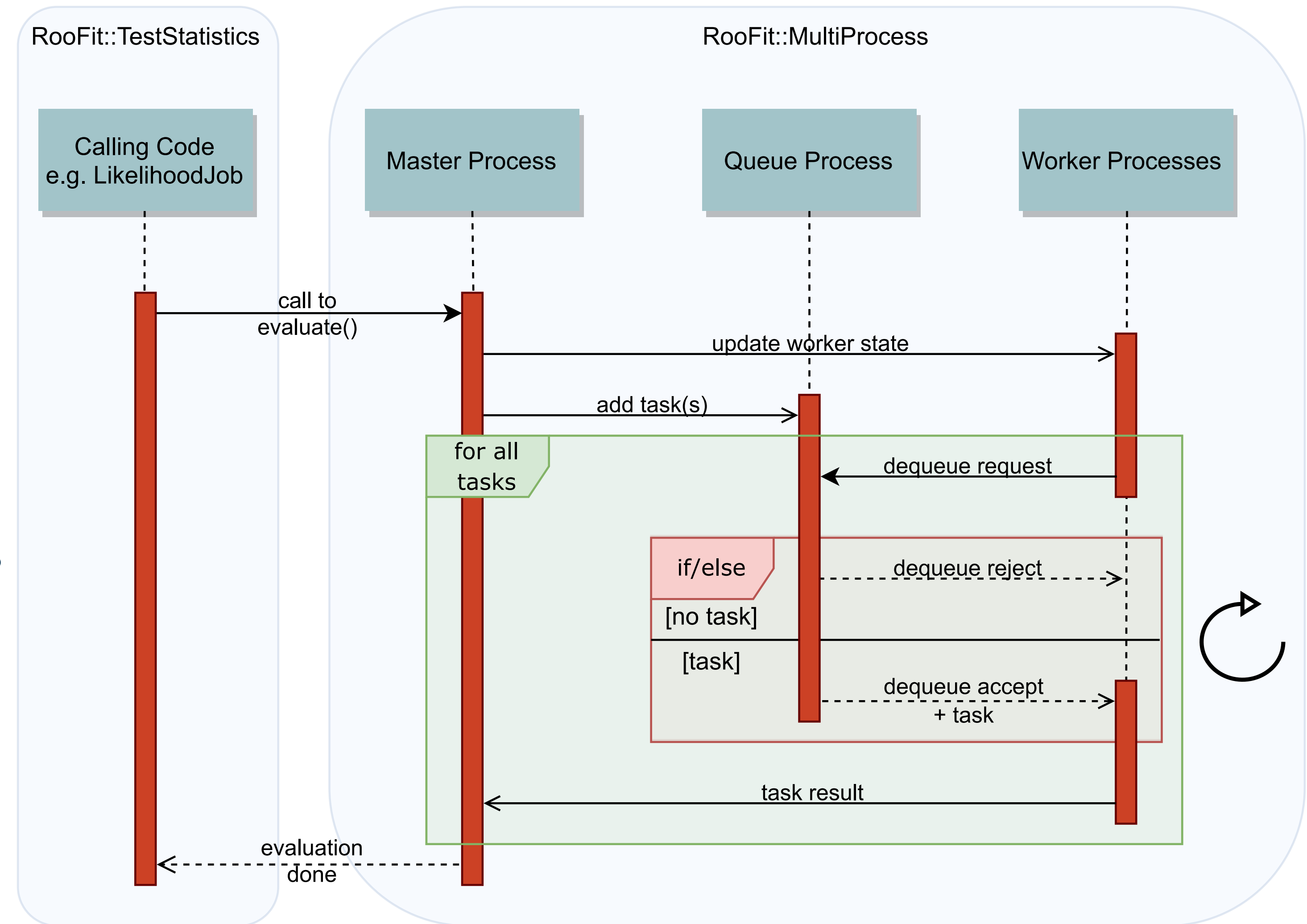
RooFit::MultiProcess knows how to interact with “Job” base classes

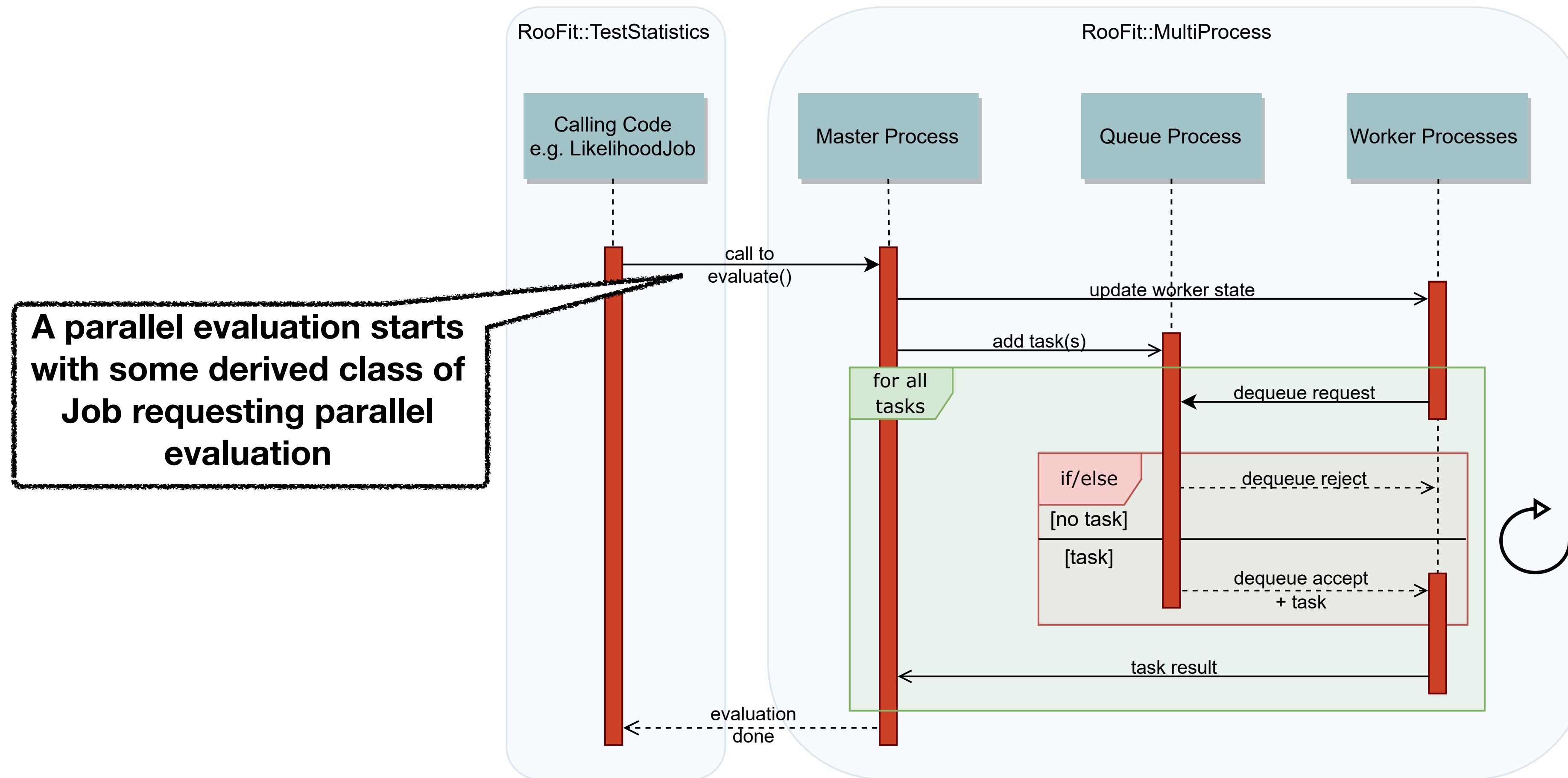


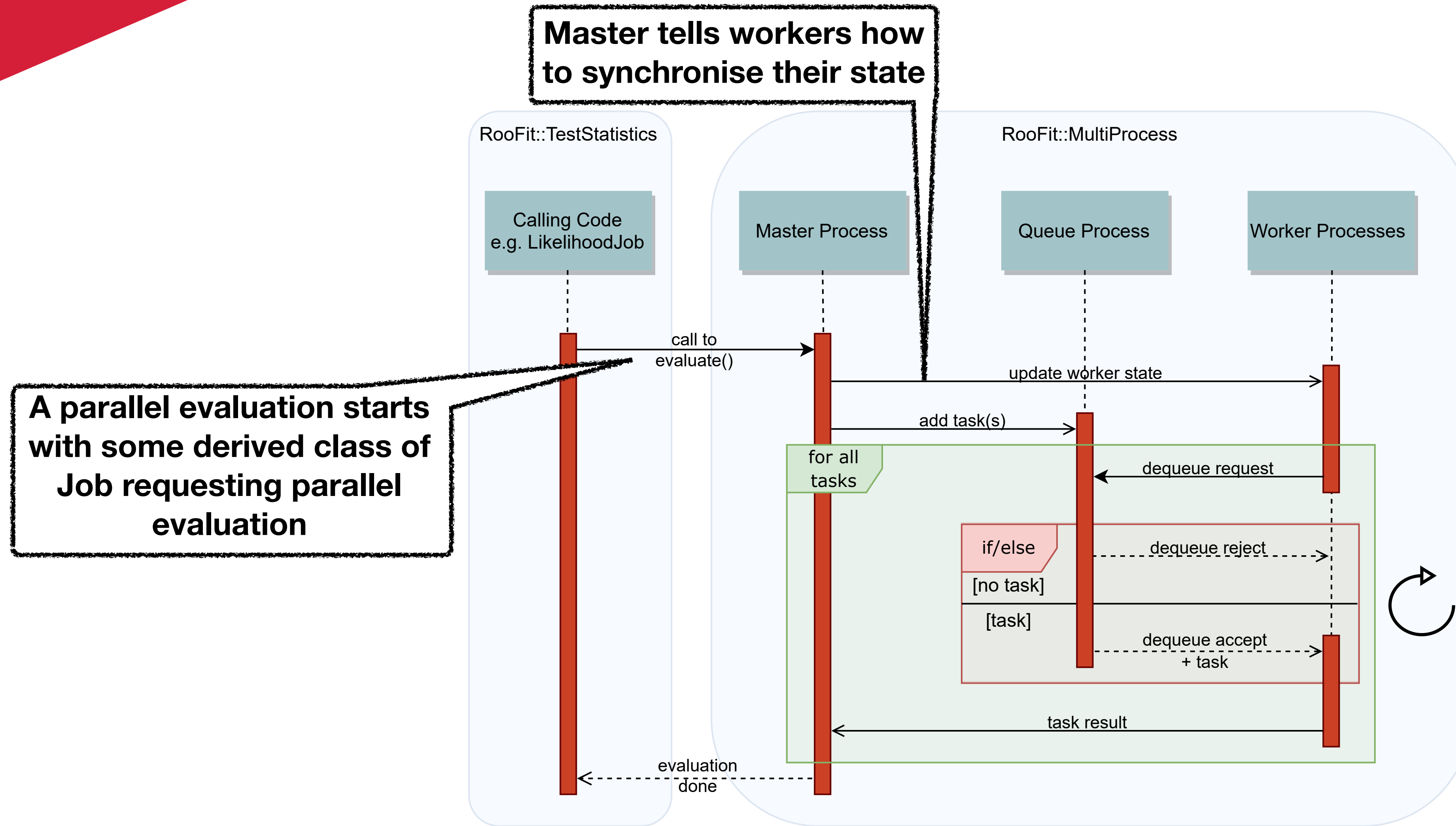
The rest of RooFit knows how to interact with likelihoods, gradients, etc...

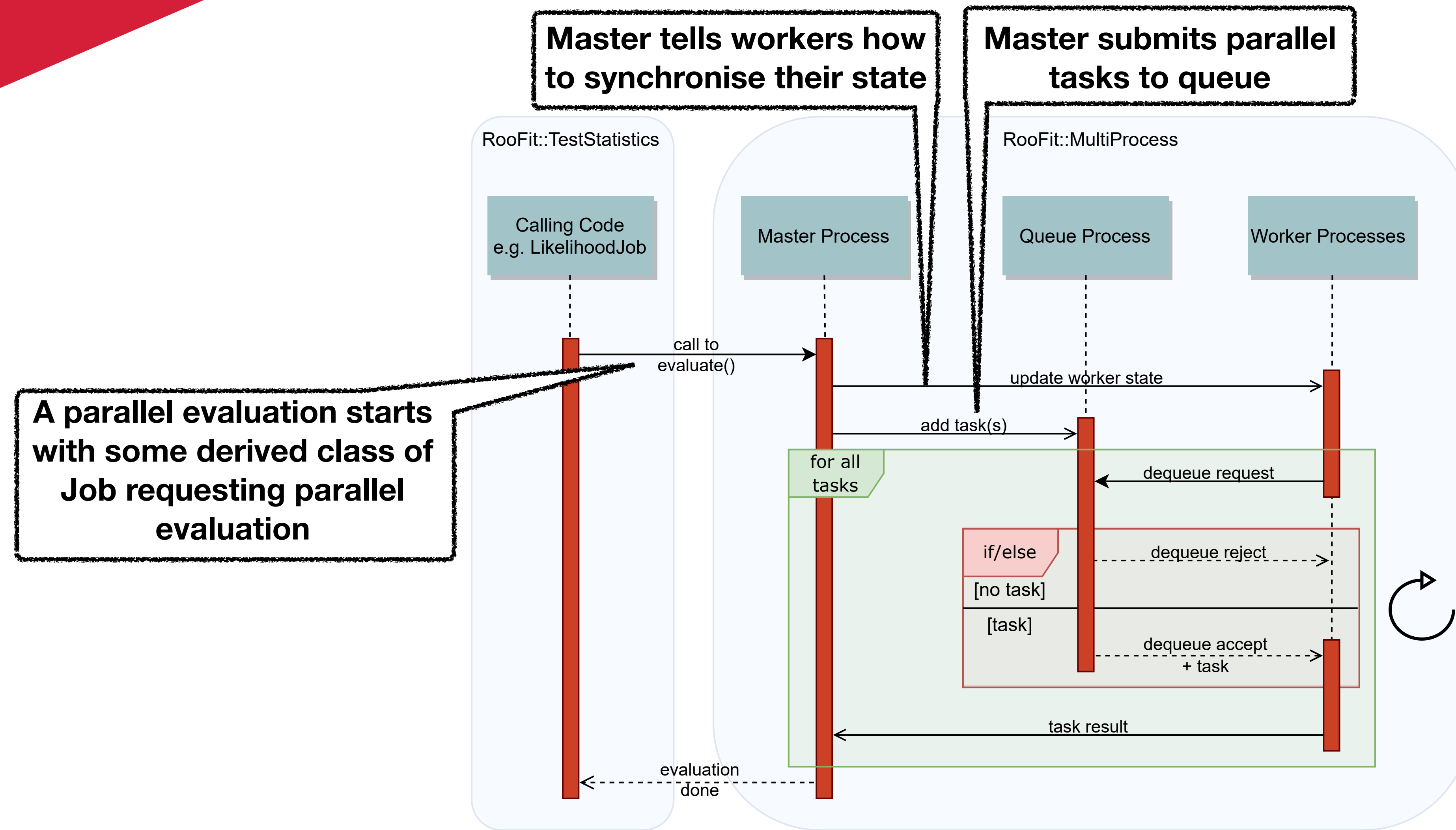
Classes that inherit from `MultiProcess::Job` thus interface multiprocessing code with the rest of RooFit

- The UML sequence diagram included on the right displays a simplified version of the `RooFit::MultiProcessing` execution flow
- Much more detailed UML diagrams of `RooFit::MultiProcessing` can be found in previous CHEP proceedings [1]

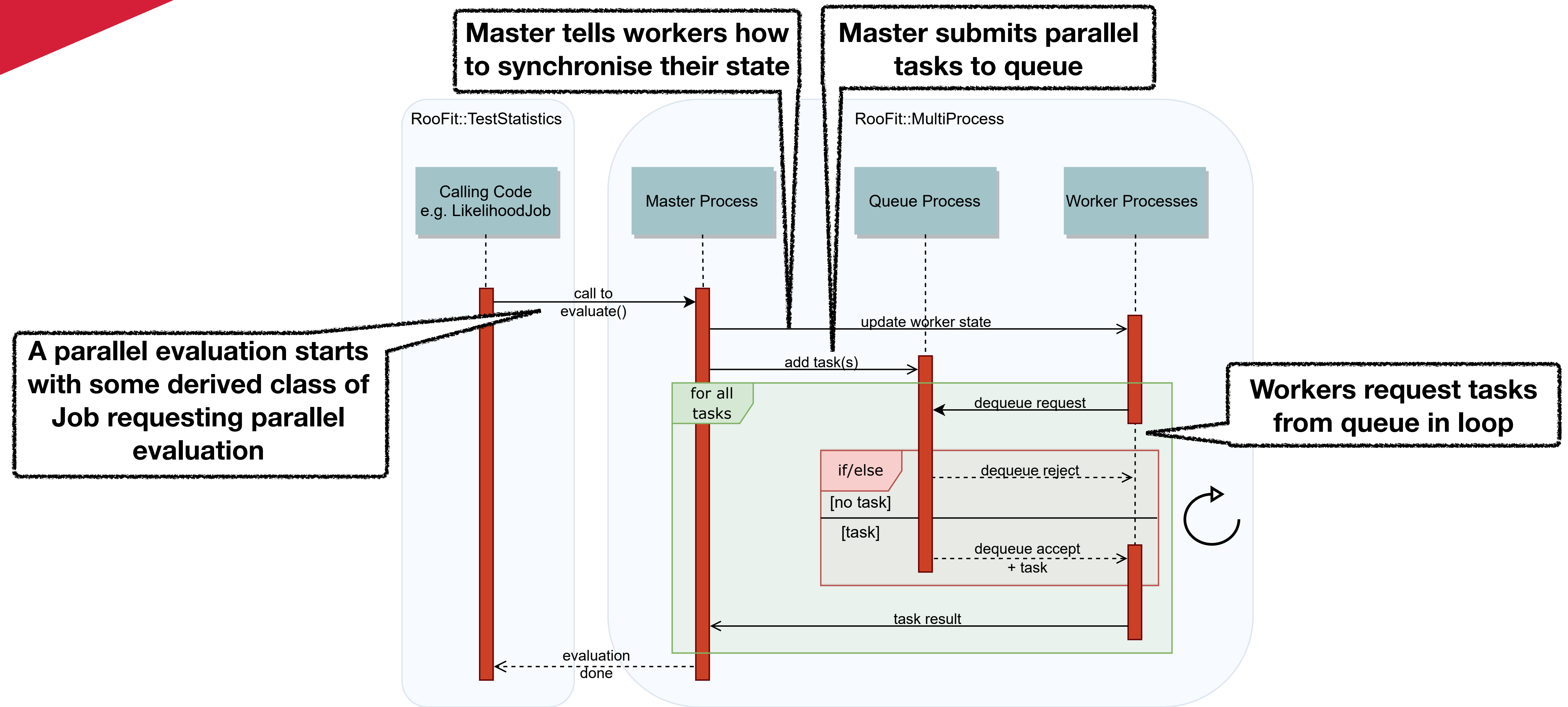




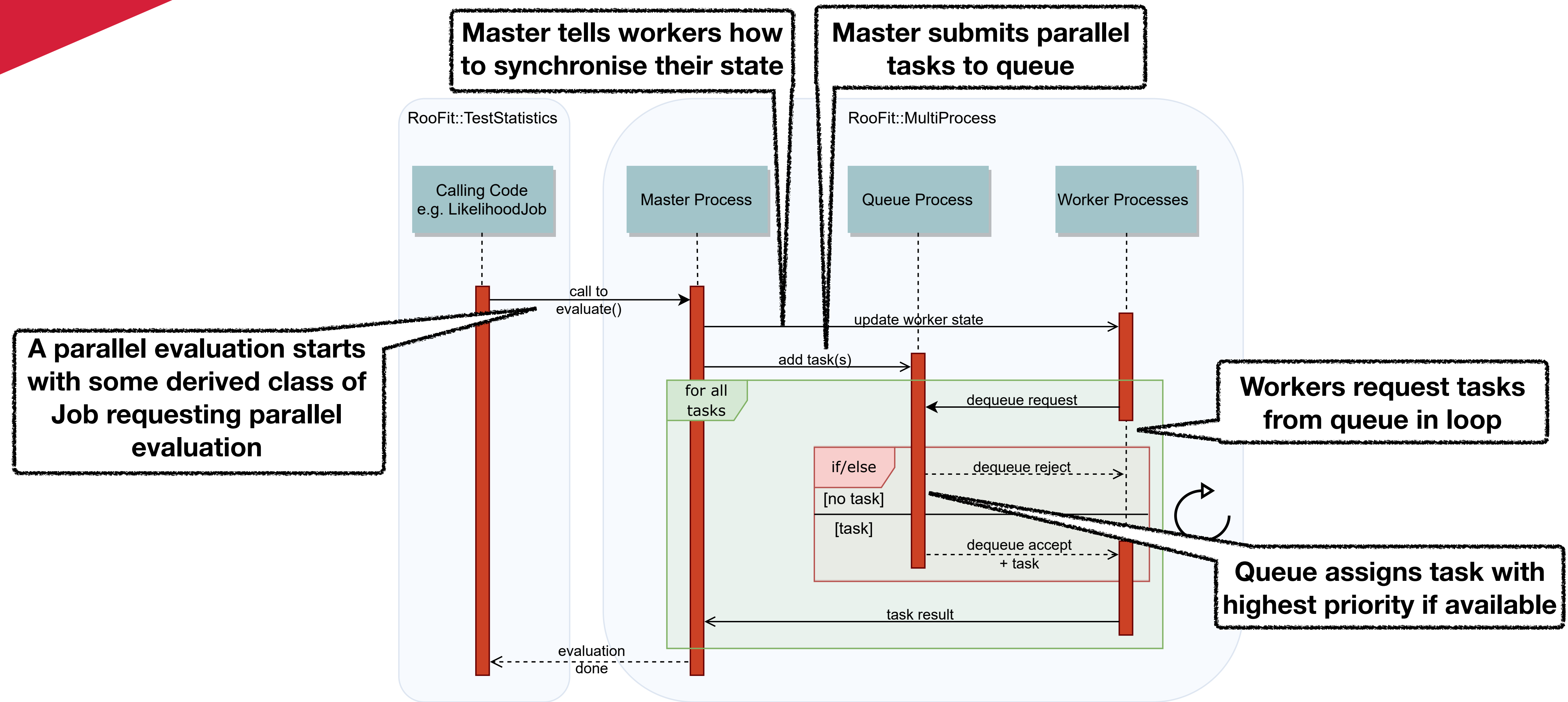




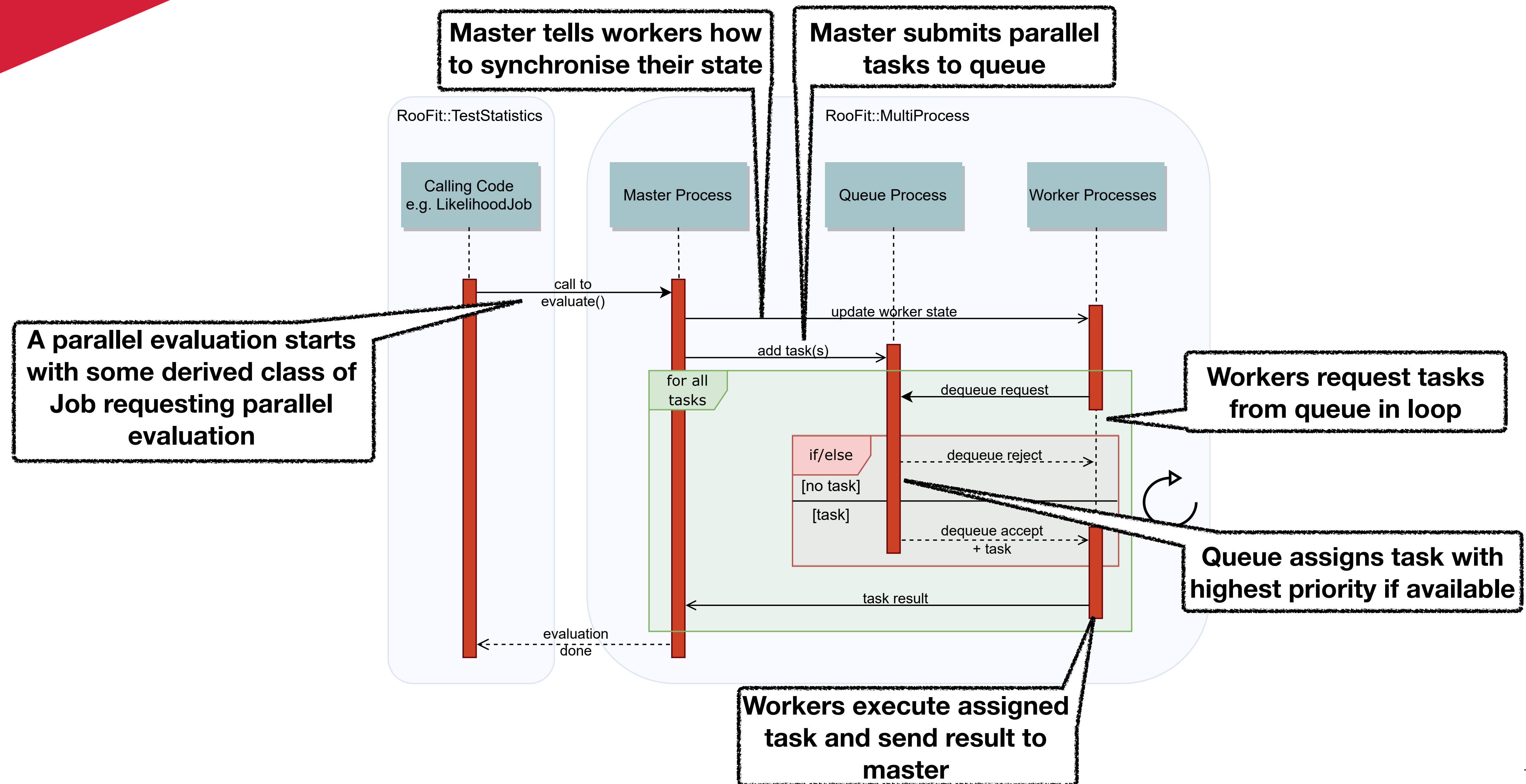
A General Parallel RooFit Framework

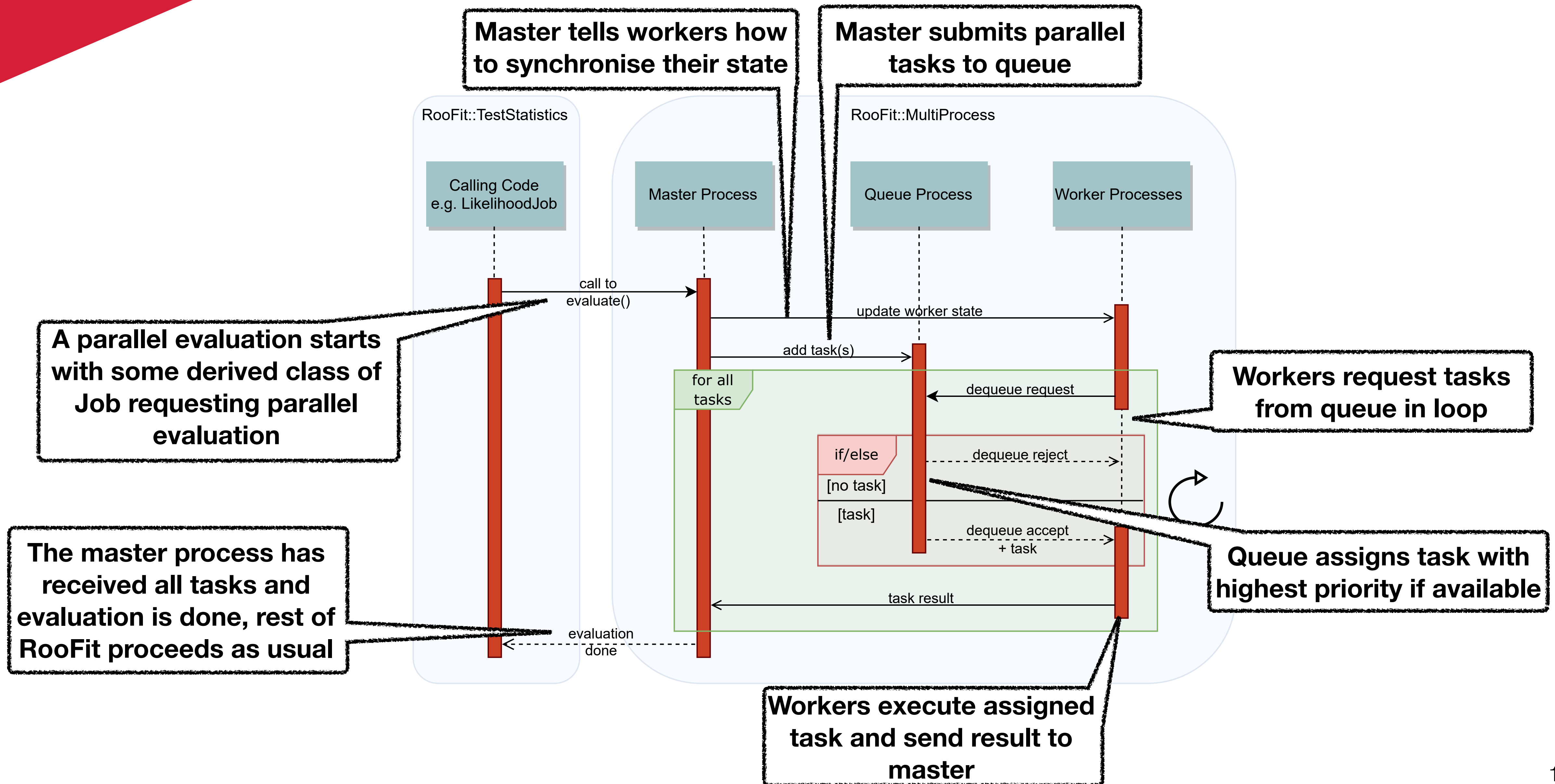


A General Parallel RooFit Framework

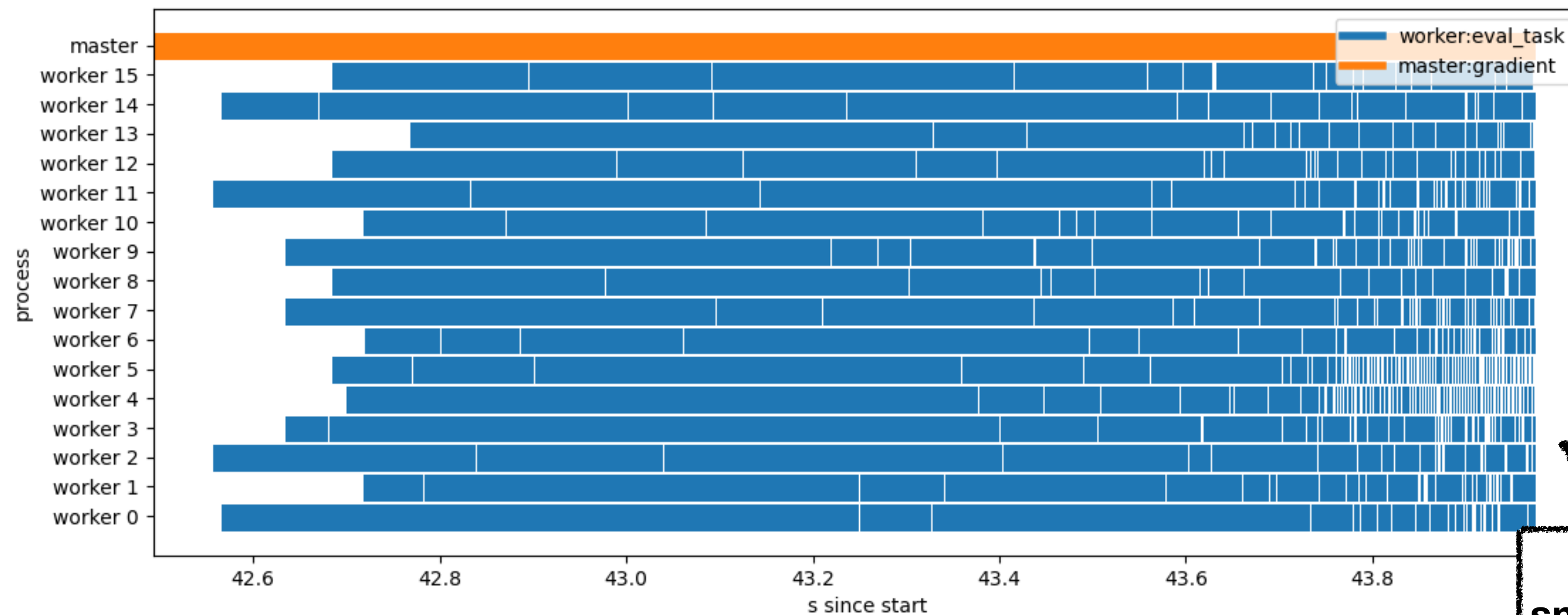


A General Parallel RooFit Framework





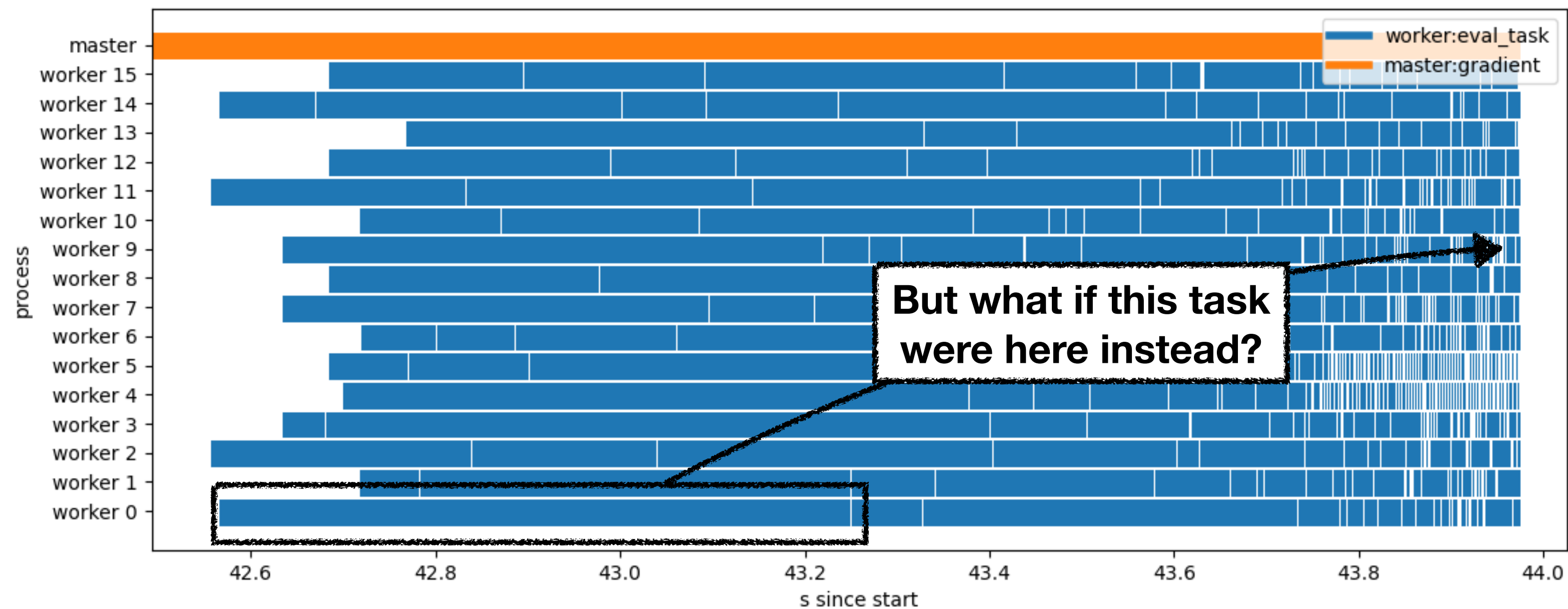
- The ordering of parallel tasks can significantly impact the total runtime of a parallel program
 - Suboptimal ordering in cases where task duration varies strongly can cause processes to idle



When ending with the smallest jobs workers do not have to wait for each other

- `RooFit::MultiProcessing` implements custom task ordering
 - Can be dynamically updated with timing information as the variable metric steps progress
 - Reduces gradient calculation time by more than 5% for 10 workers “for free”

- The ordering of parallel tasks can significantly impact the total runtime of a parallel program
 - Suboptimal ordering in cases where task duration varies strongly can cause processes to idle



- `RooFit::MultiProcessing` implements custom task ordering
 - Can be dynamically updated with timing information as the variable metric steps progress
 - Reduces gradient calculation time by more than 5% for 10 workers “for free”

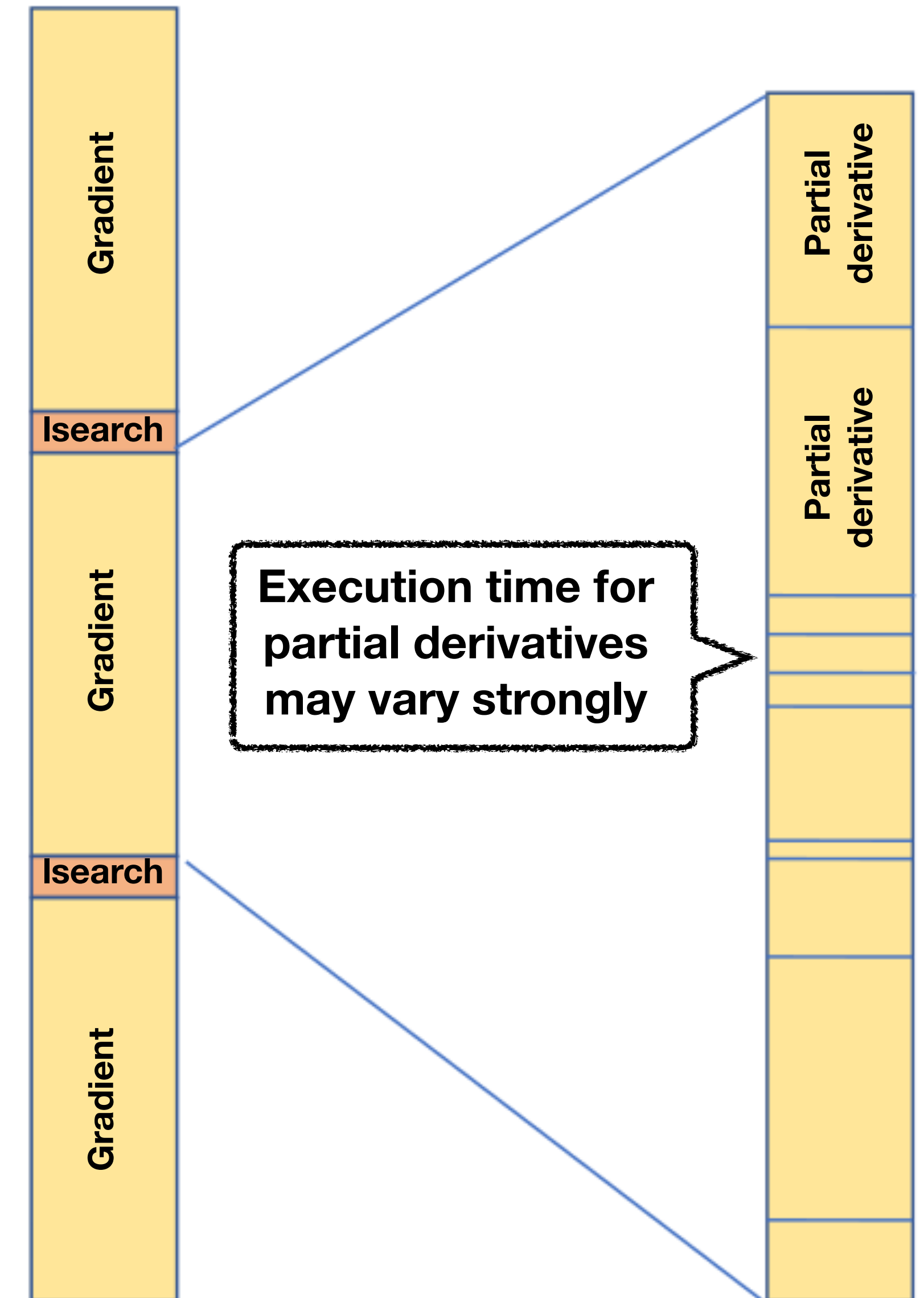
Minimisation with Gradient and Likelihood Parallelisation

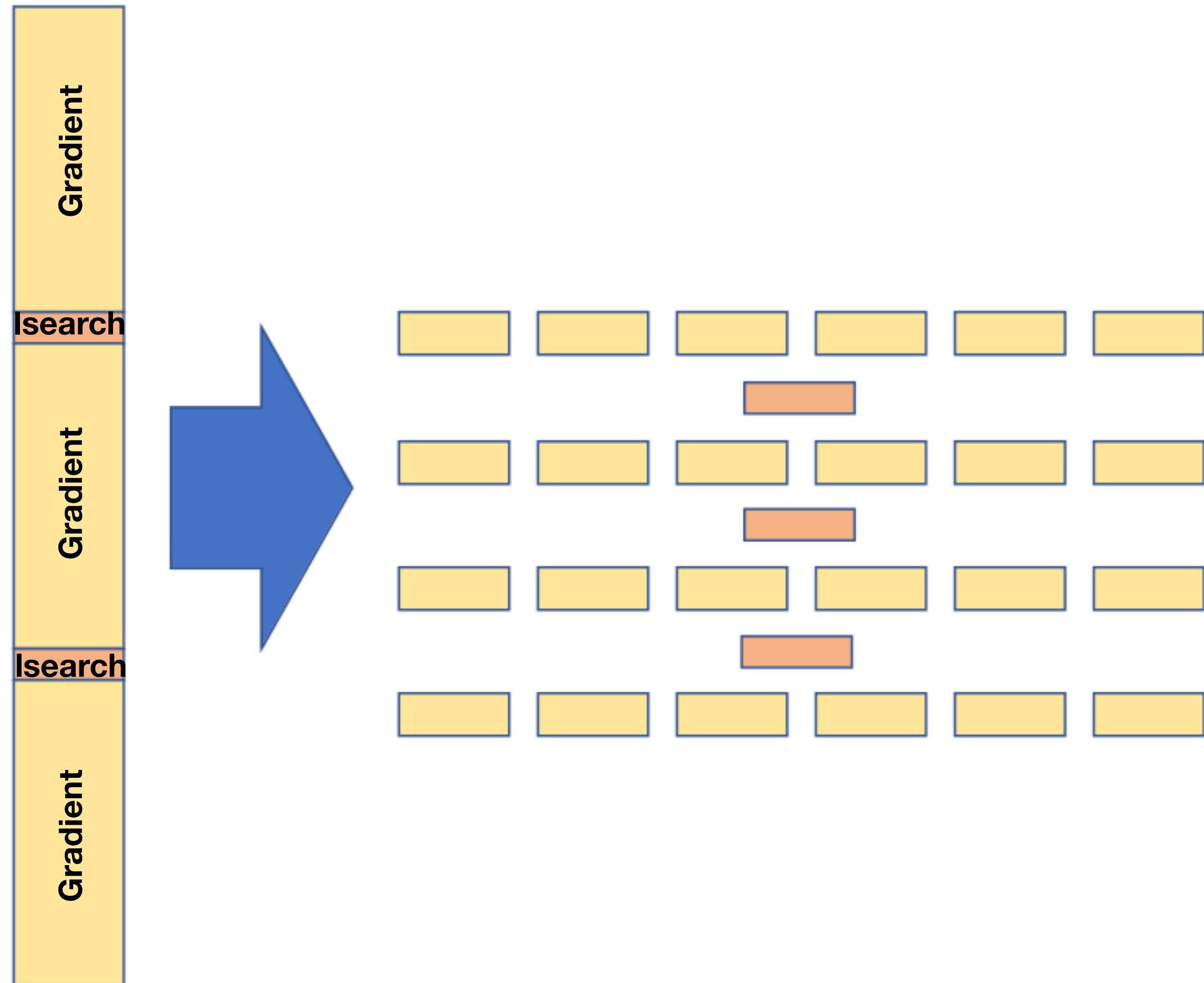
- The principle behind most minimisation routines consists of

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda \mathbf{p} \text{ such that } f(\mathbf{x}_{i+1}) < f(\mathbf{x}_i)$$

until some stopping condition is satisfied

- For Minuit2, the minimisation routine that RooFit uses, the following holds
 - \mathbf{p} is the step direction, determined by the variable metric method, the most expensive part of which is the **calculation of the gradient ($O(N)$ likelihood evals)**
 - λ is the step size in the given direction, determined by a line search step, the most expensive part of which is the **evaluation of the full likelihood ($O(3)$ likelihood evals)**





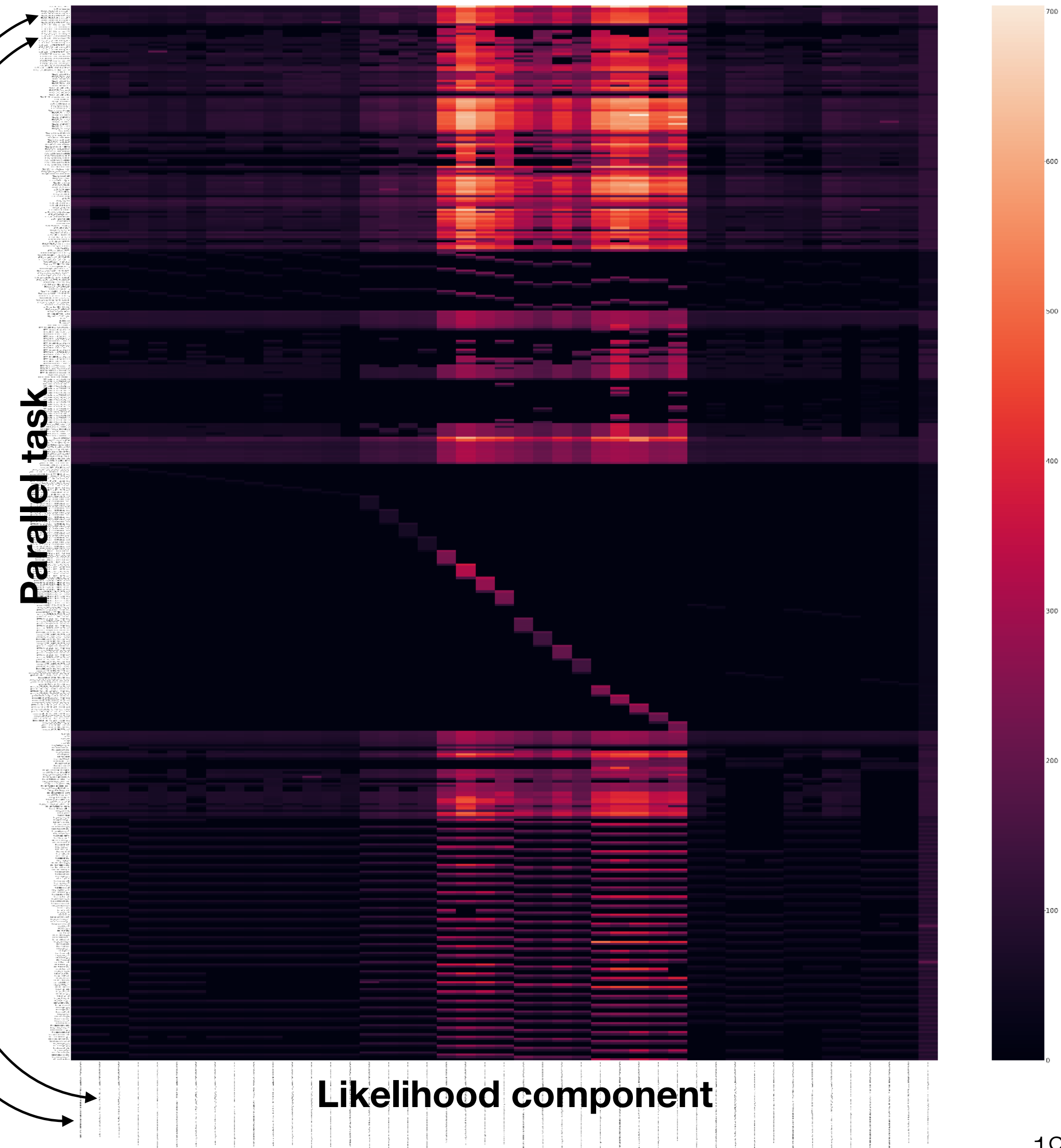
- `RooFit::TestStatistics` splits the gradient into individual partial derivative tasks
- The task (partial derivatives) sizes may vary strongly due to
 - Most components only being dependent on subset of parameters, thus not all components need evaluation for every partial derivative
 - Varying likelihood component calculation complexity
- Dynamic load balancing is crucial and is currently addressed by
 - Work stealing algorithm
 - Task ordering by duration

- The heat map on the right shows a single gradient calculation, distributed across multiple workers
 - This particular gradient took place when fitting a $H \rightarrow WW$ workspace

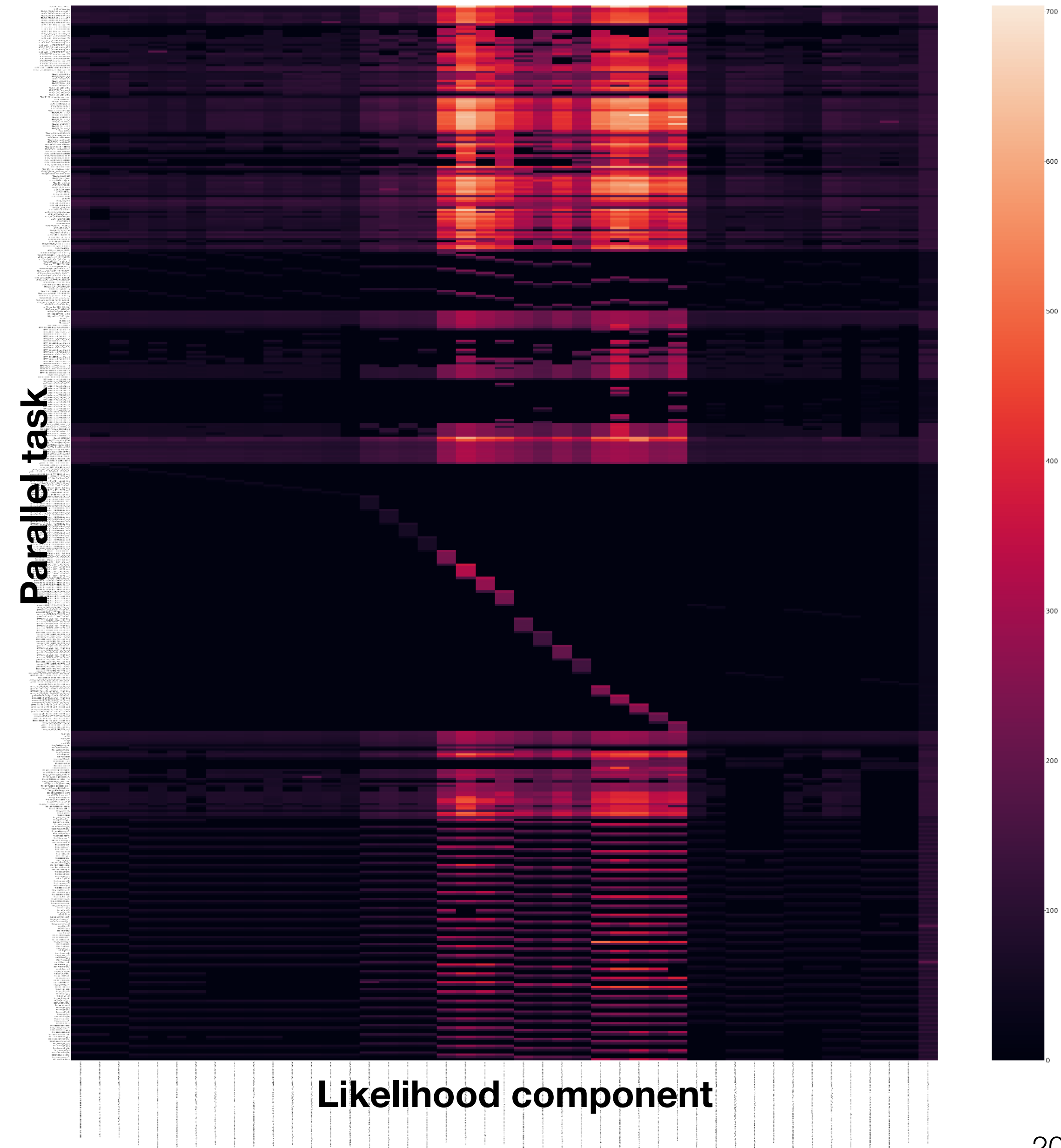
- Remember the gradient:

$$\nabla NLL = \frac{\partial NLL}{\partial \mathbf{x}} = \begin{bmatrix} \partial_{x_1} NLL(comp.1) + \partial_{x_1} NLL(comp.2) + \dots \\ \partial_{x_2} NLL(comp.1) + \partial_{x_2} NLL(comp.2) + \dots \\ \vdots \end{bmatrix}$$

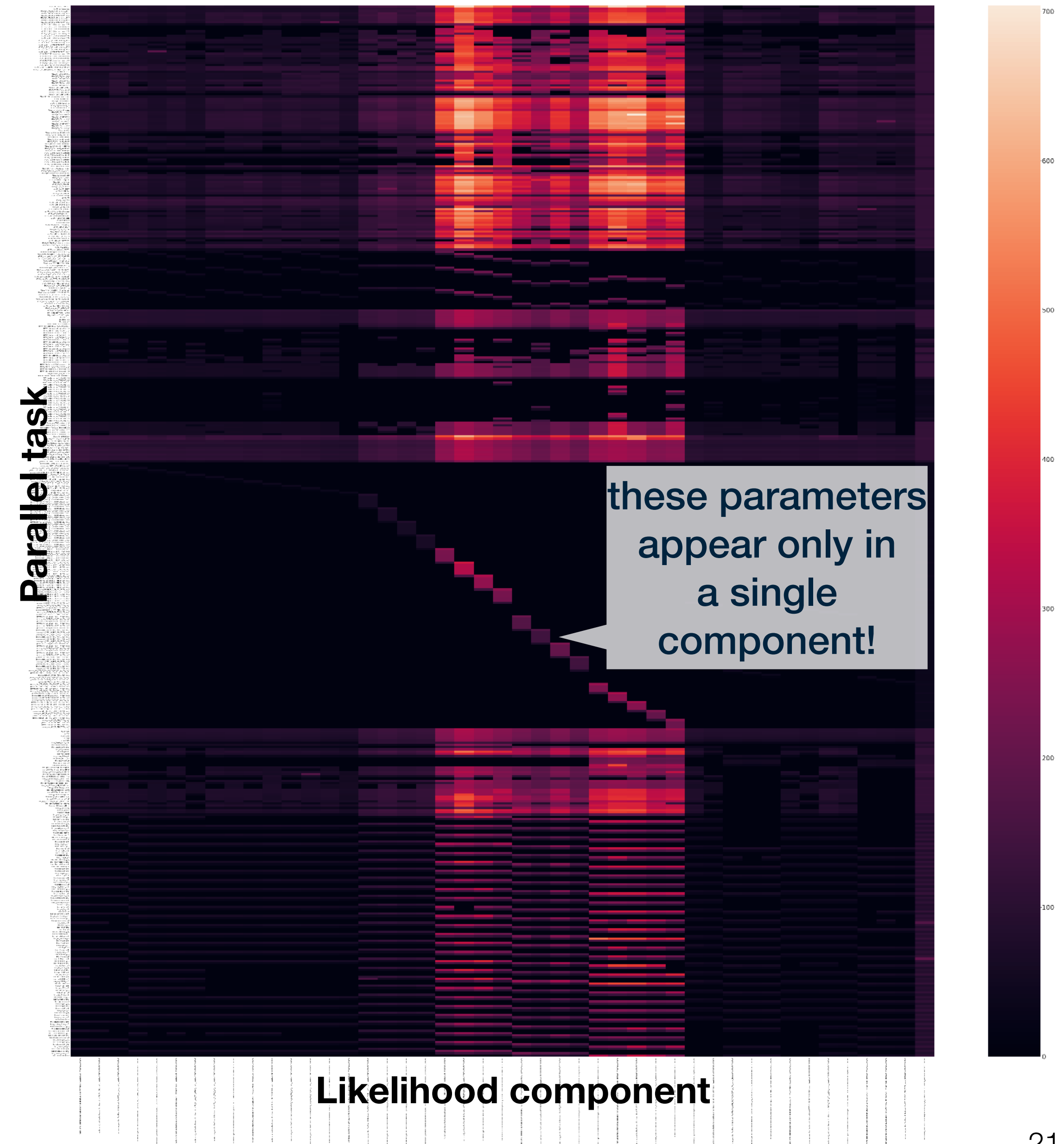
- The x-axis indicates the likelihood component
- The y-axis indicates a parallel task, in this case split by partial derivatives



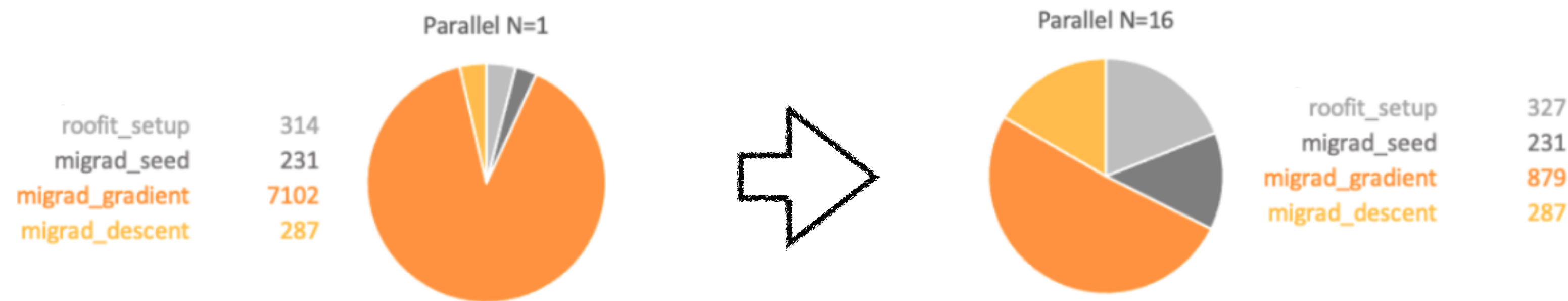
- Not all parameters present in all likelihood components
 - If this is the case, no evaluation is necessary and the result is returned immediately
 - Explains the black regions in heatmap
- Benchmarking tools now available in RooFit
 - `TimingAnalysis` argument in `RooMinimizer` enables profiling
 - `RooFit::MultiProcess::HeatmapAnalyzer()` to create a heatmap



- Not all parameters present in all likelihood components
 - If this is the case, no evaluation is necessary and the result is returned immediately
 - Explains the black regions in heatmap
- Benchmarking tools now available in RooFit
 - `TimingAnalysis` argument in `RooMinimizer` enables profiling
 - `RooFit::MultiProcess::HeatmapAnalyzer()` to create a heatmap

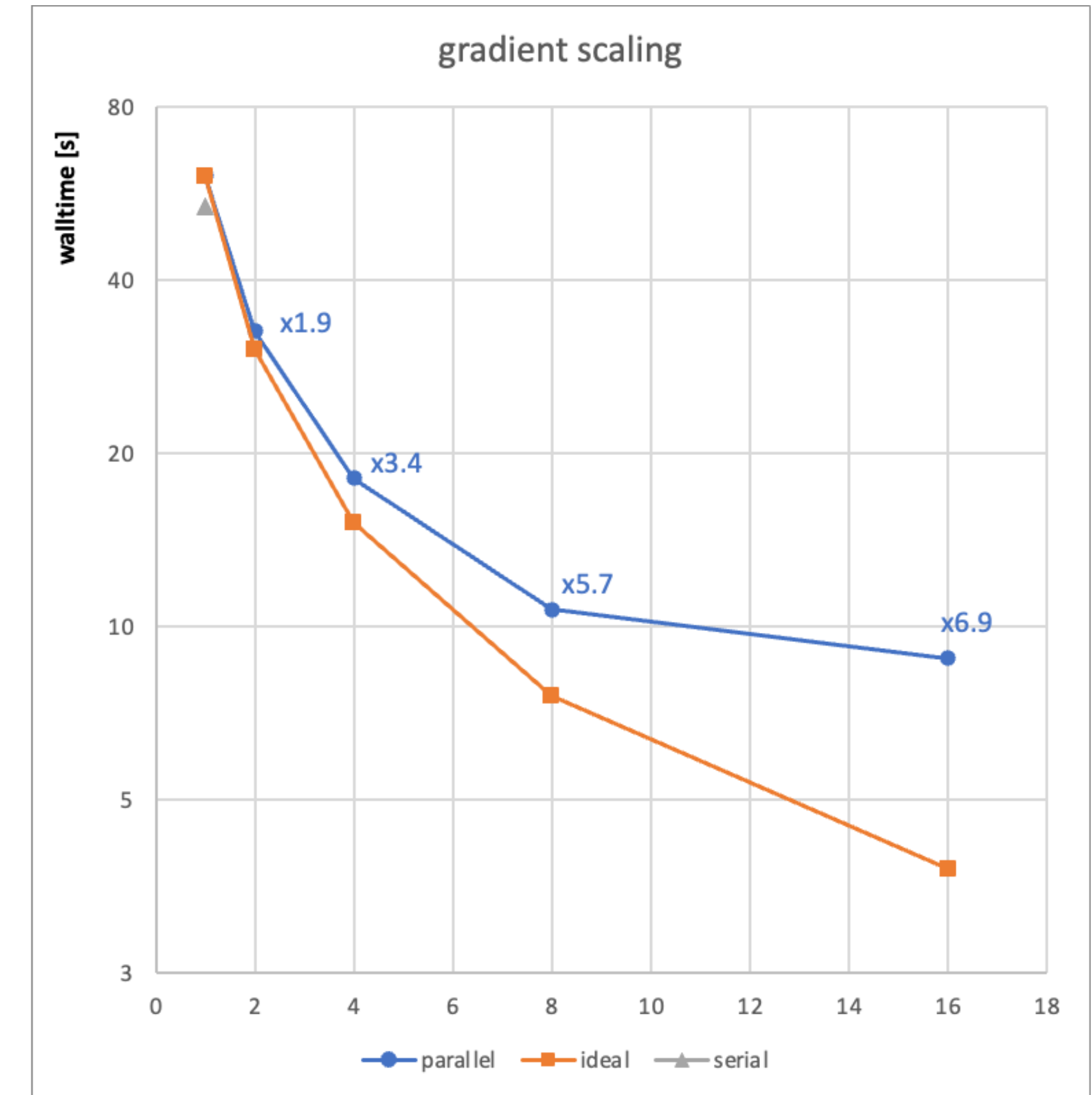


- In some cases, evaluation of the likelihood can be the bottleneck, for example in the calculation of the line search step
 - During the line search step all parameters are typically changed two or three times, requiring an evaluation of all components of the likelihood
 - With the gradient sufficiently optimised, this can become the bottleneck for an entire fit



- `RooFit::TestStatistics` has two options for splitting likelihood evaluation into tasks
 - By events: each task is defined by an event range to execute
 - By components: each task is defined by a set of components to execute

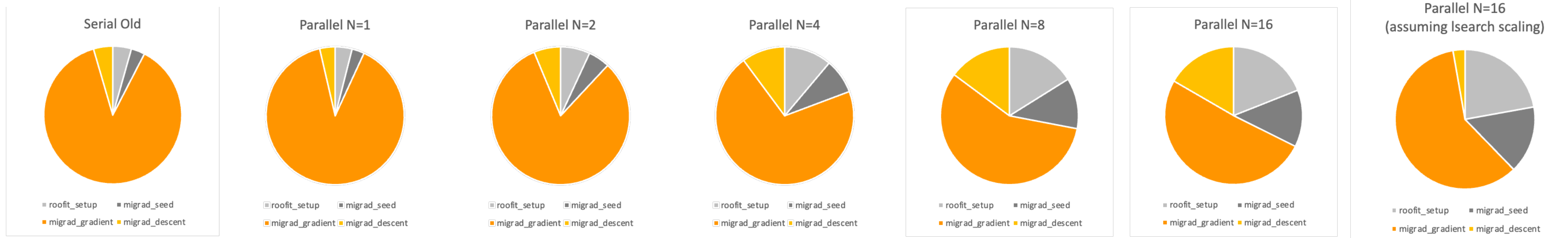
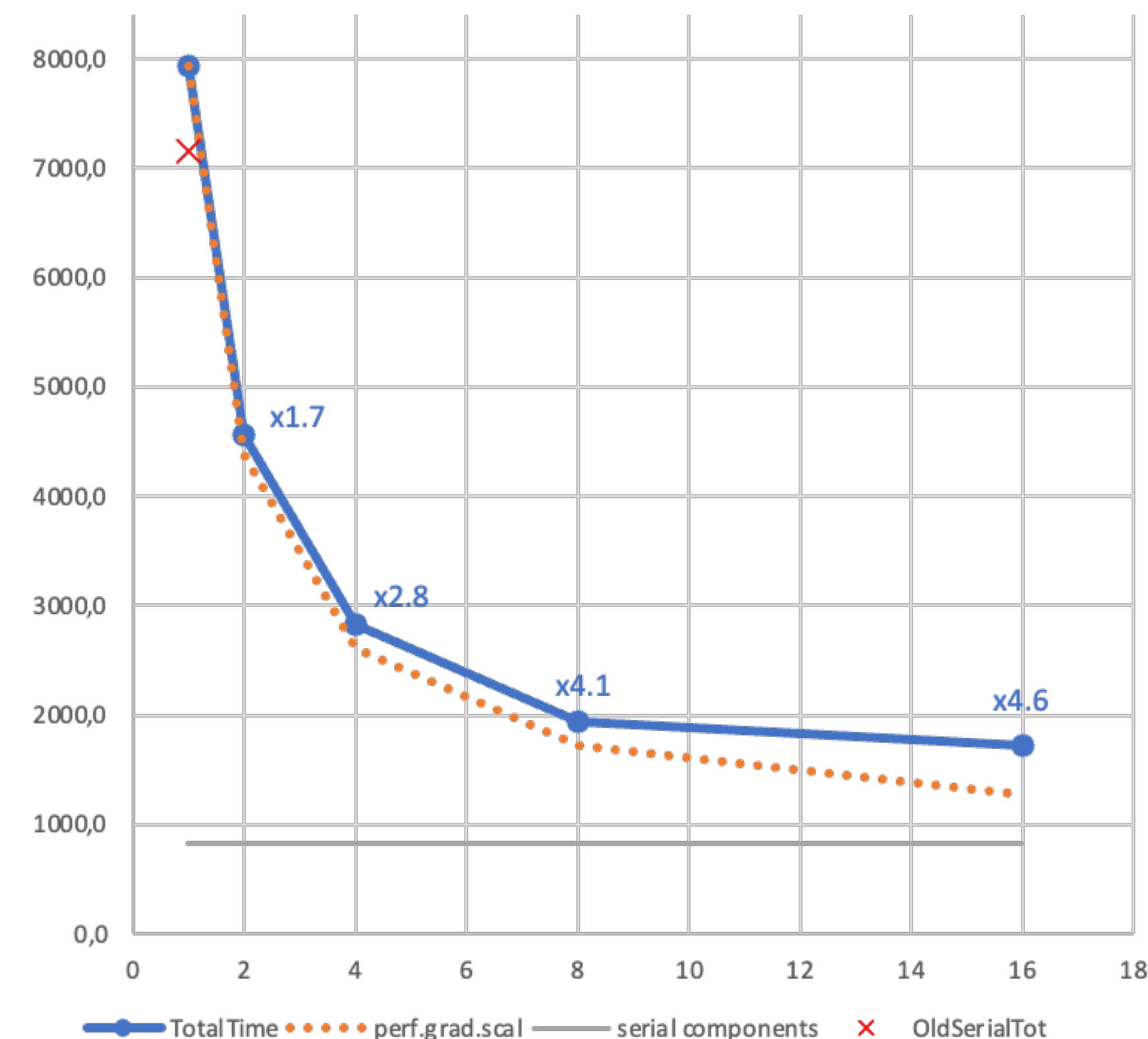
Results



- Used recent Higgs combination workspace produced for 10 year Higgs anniversary paper [2]
- The line search is work in progress, gradient can be used out of the box in ROOT 6.28
 - For the line search timings $H \rightarrow \gamma\gamma$ was removed from the combination workspace

- With gradient parallelisation the achieved speedup with 16 workers is 4.6, including all serial components
 - This brings the walltime down from 2 hours and 12 minutes to 29 minutes
 - At that point, nearly half of the walltime is spent in serial parts
- With line search parallelisation fully integrated we can **reasonably expect** to reach a total speedup of 5.3
 - Would bring walltime down to 25 minutes

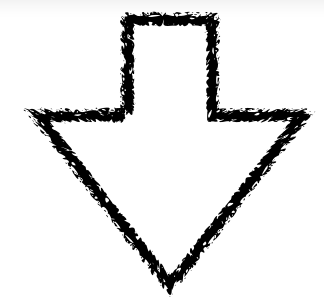
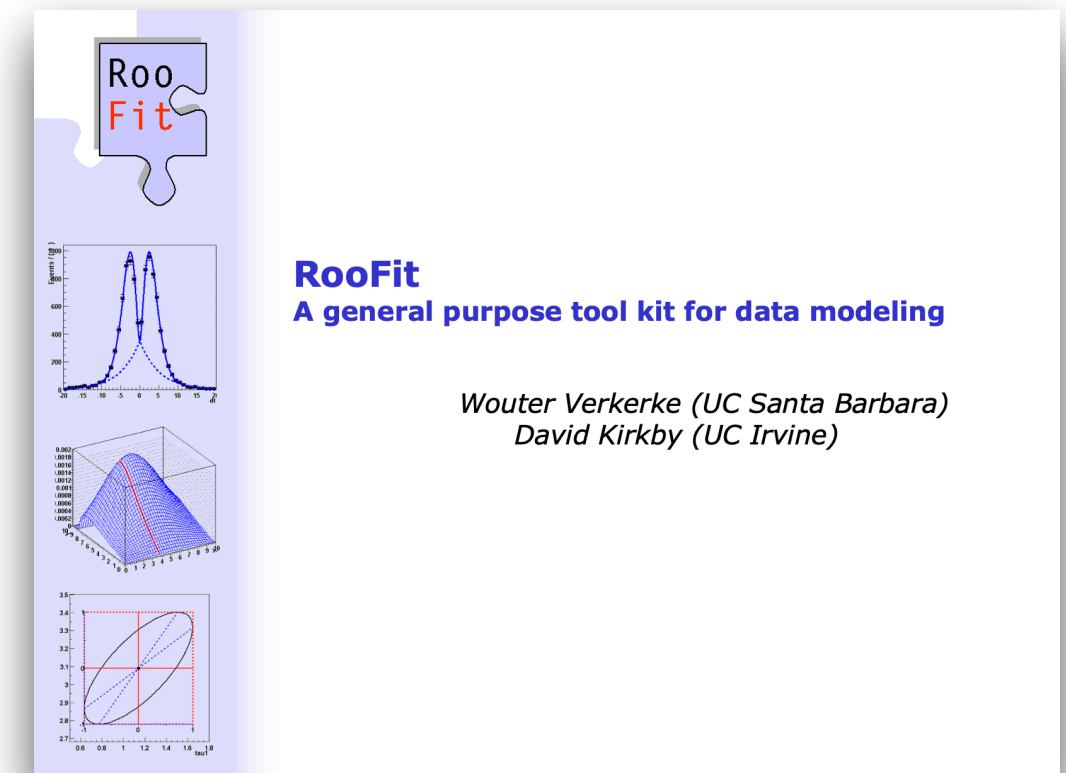
Total Fit Time scaling



Conclusions

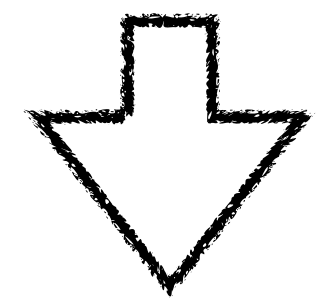
- Higgs combination fits and the future hi-lumi LHC pose significant challenges to high energy physics analysis software
- RooFit addresses these challenges through improvements in multiple directions
 - Automatic differentiation (Garima's talk!)
 - Batched computations and vectorisation (Jonas' talk!)
 - Multiprocessing
- Consolidation of these efforts is an important next step on the agenda
 - For example, multiprocessing and batched computations optimise at a different level and could be used simultaneously

CHEP 2003

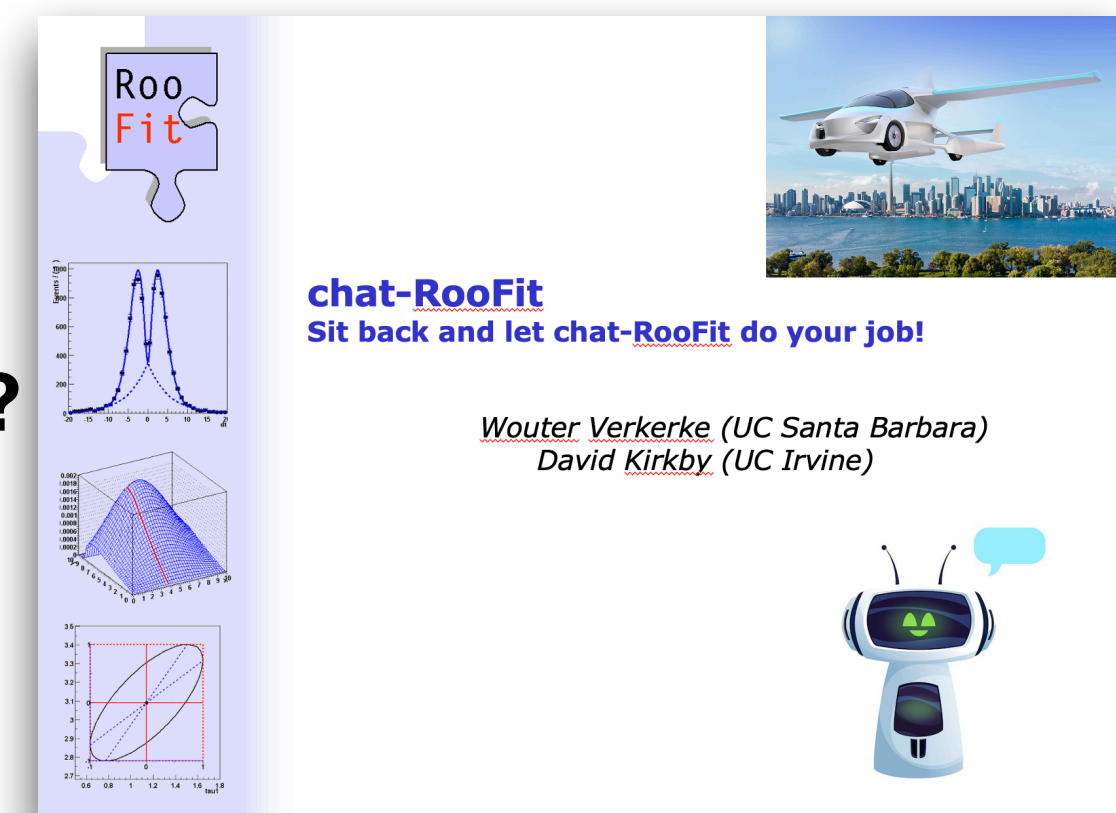


CHEP 2023

You are here!



CHEP 2043 ?

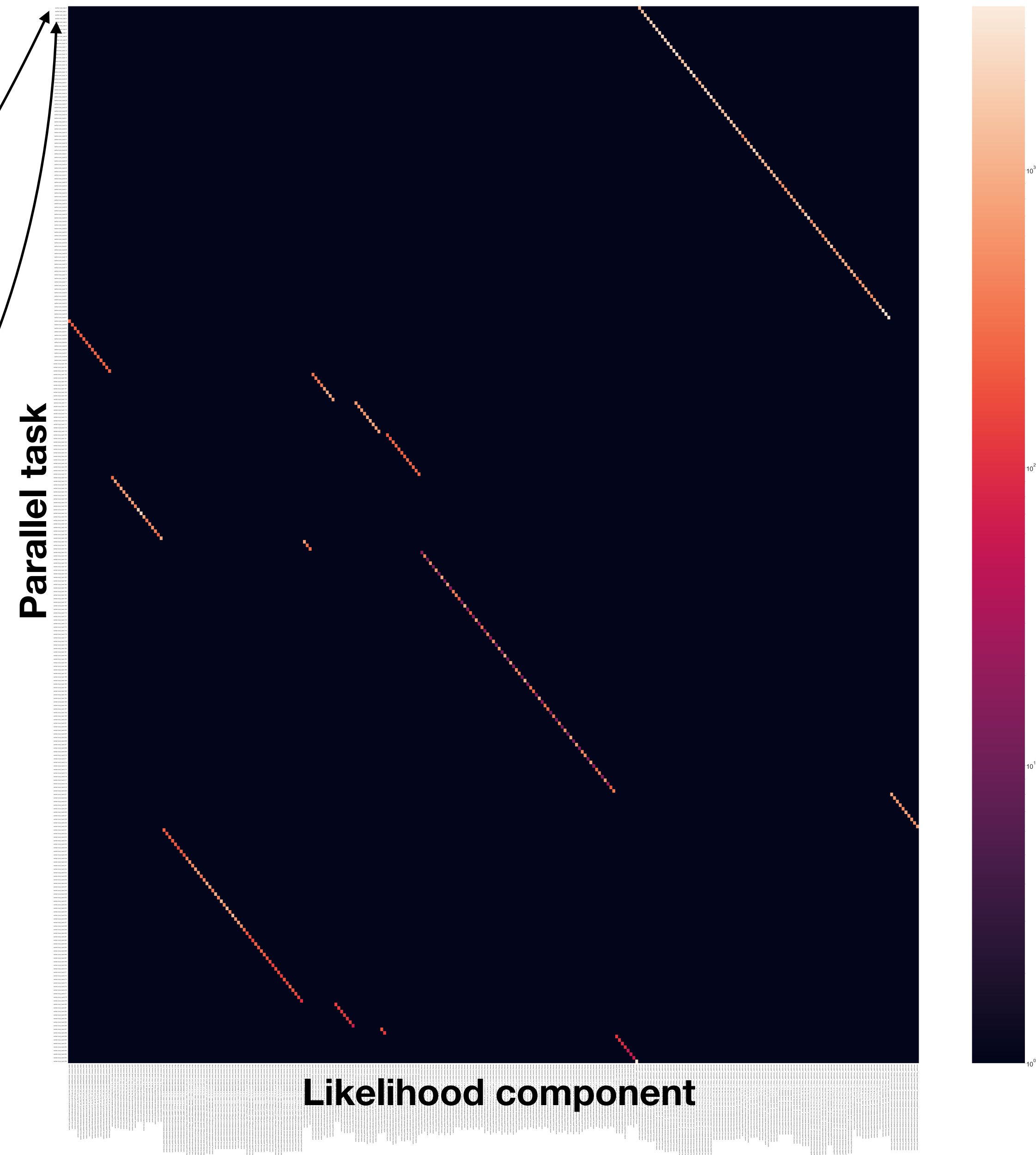


Backup

- This heat map displays a single line search evaluation

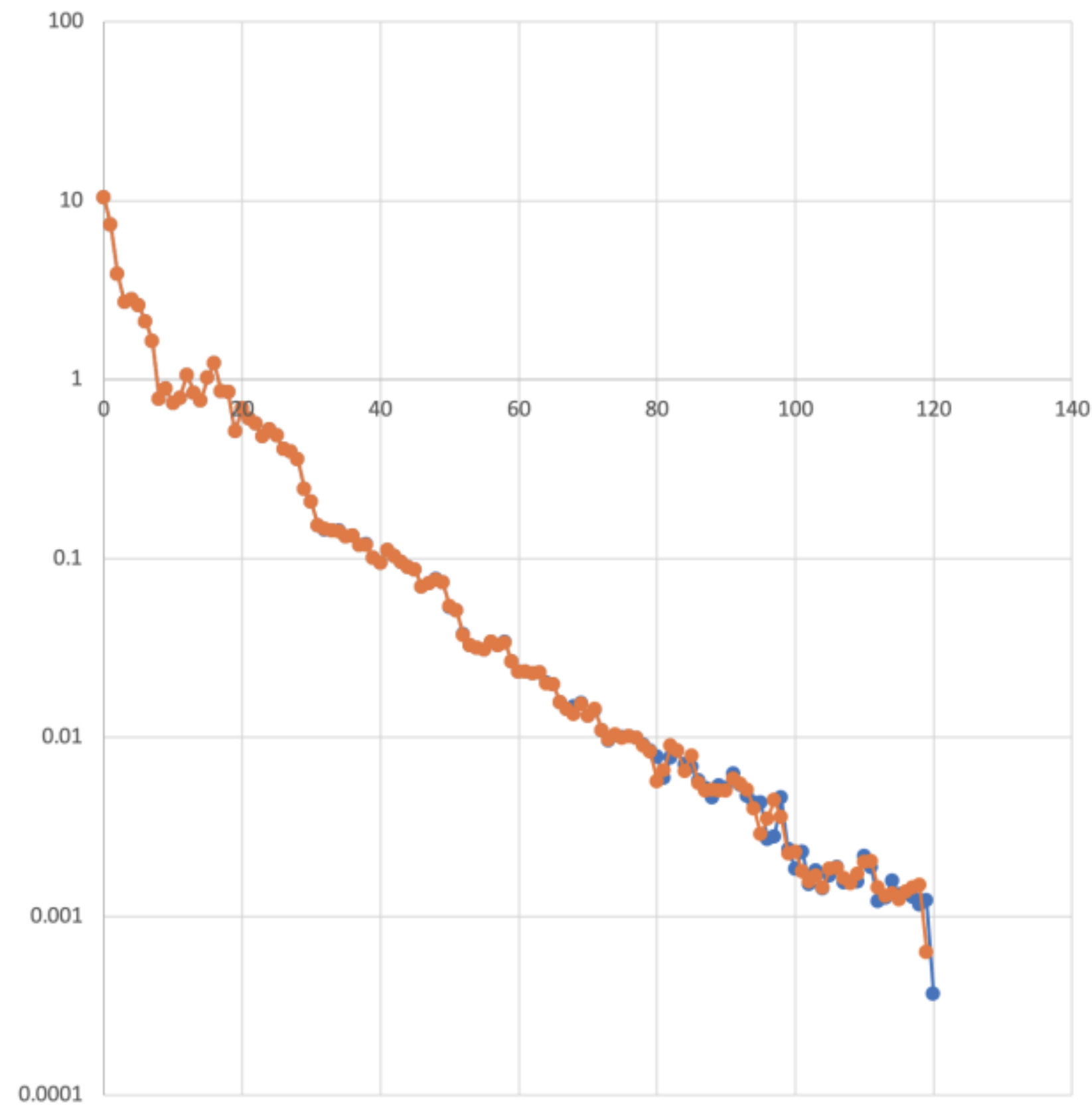
$$\begin{aligned} -\log L(\theta | \mathbf{x}) &= -\log \prod_{i=0}^N p_i(\mathbf{x} | \theta) \quad N \equiv N_{\text{components}} \\ &= -\sum_{i=0}^N \log(p_i(\mathbf{x} | \theta)) \\ &= -\log(p_1(\mathbf{x} | \theta)) - \log(p_2(\mathbf{x} | \theta)) - \dots \end{aligned}$$

- This line search was parallelised by components
 - As such, every parallel task (y-axis) does one component evaluation
 - This explains the diagonal lines



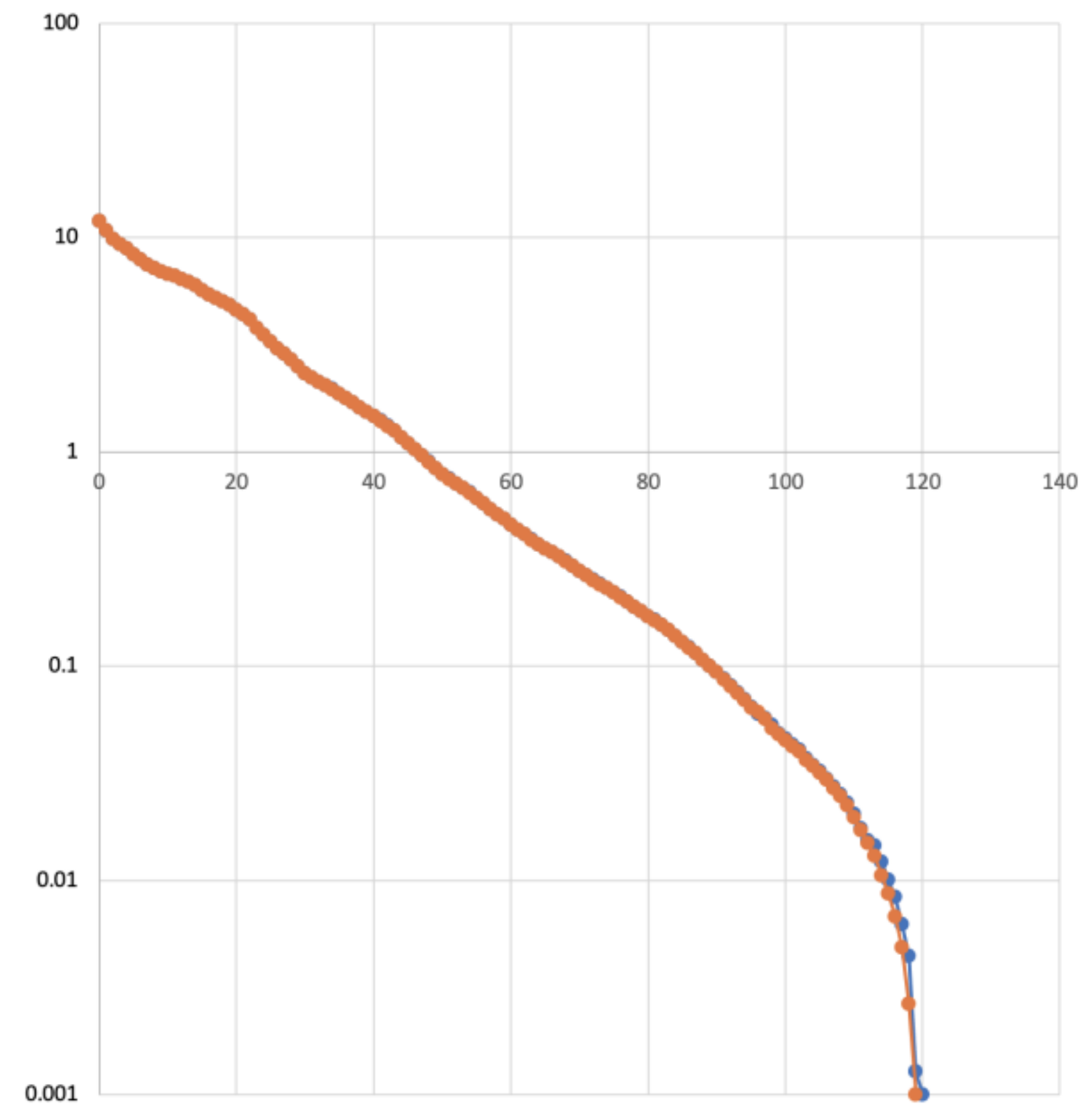
Hcomb workspace (120 VM steps, Npar=3105, Ncomp=334)

EDM vs VariableMetric step



$-\log(L)$ vs VariableMetric step

(L offset such that minimum is by definition at 0.001)



Convergence

99% of the 3105 parameters agree within 0.1% of estimated uncertainty

All parameters agree within 1% of the estimate uncertainty