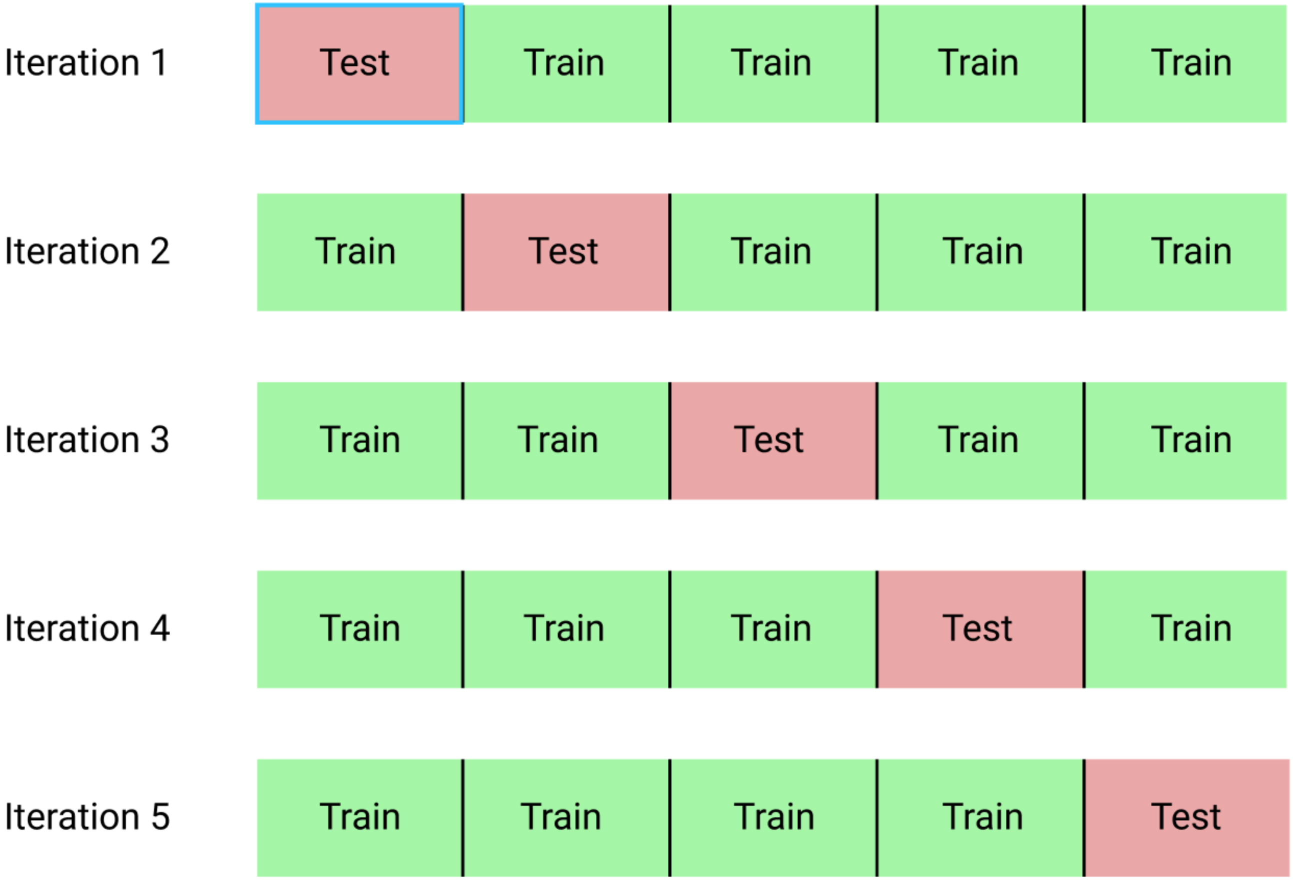
TMVA Parallelization

- There are some validation and optimization algorithms in TMVA such as:
  - Cross Validation
  - Variable Importance
  - Hyper Parameter Optimization

Cross Validation

- CV is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set.
- The most common type of CV is k-Fold Cross Validation.

K-Fold Cross Validation



Iteration 1	Test	Train	Train	Train	Train
Iteration 2	Train	Test	Train	Train	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Train	Train	Test	Train
Iteration 5	Train	Train	Train	Train	Test

Variable Importance

- Variable Importance algorithm measures importance of predictor variables.
- In TMVA, there are 3 different types of VI:
  - Short
  - All
  - Random

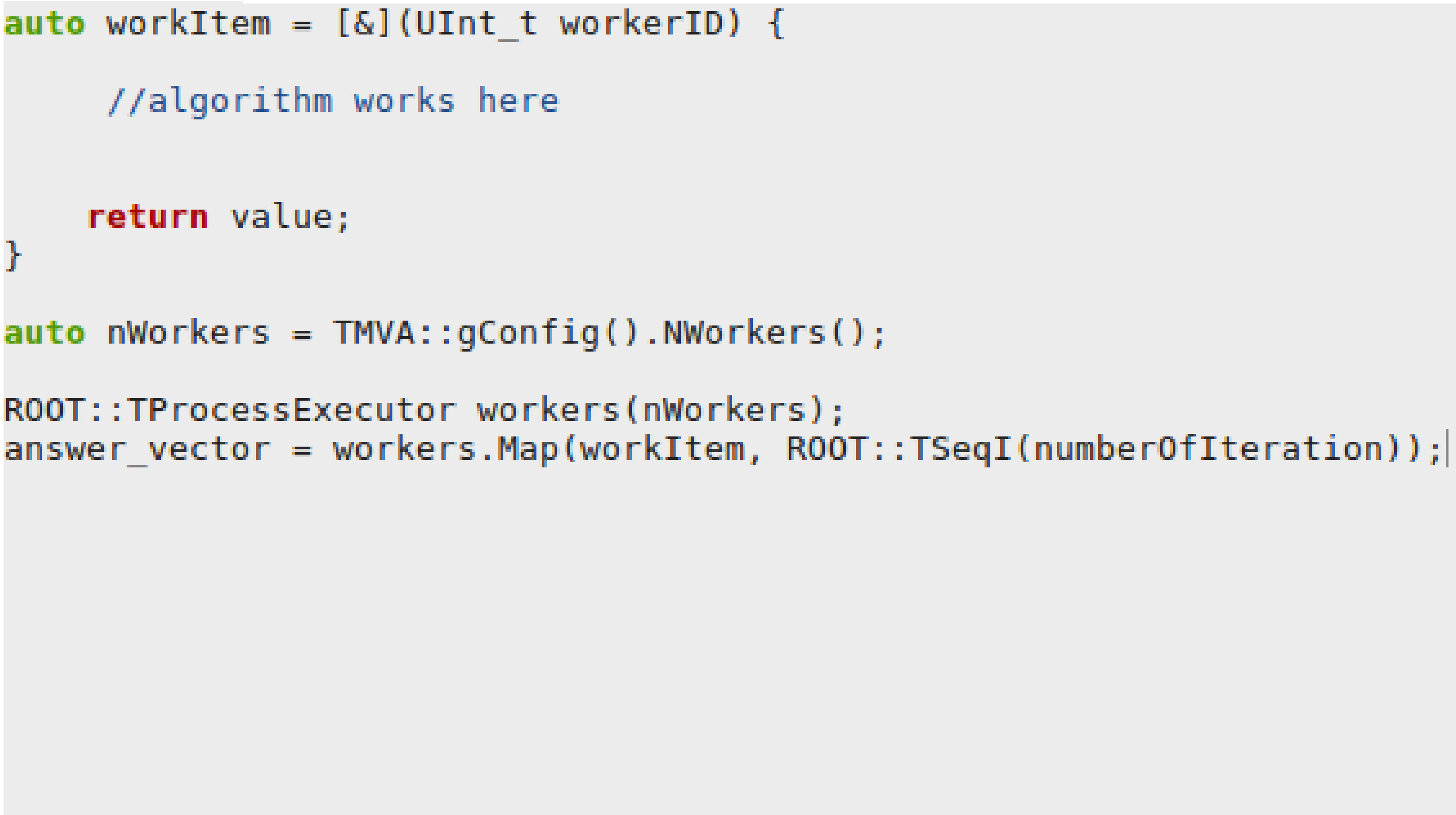
Hyper Parameter Optimization

- Hyperparameter Optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm.
- The most common HPO algorithms:
  - Grid Search
  - Random Search

- But, there are execution time problems.
- So, we must apply multiprocessing to reduce this execution time of algorithms.
- I implemented Multiprocessing by using TprocessExecutor class.

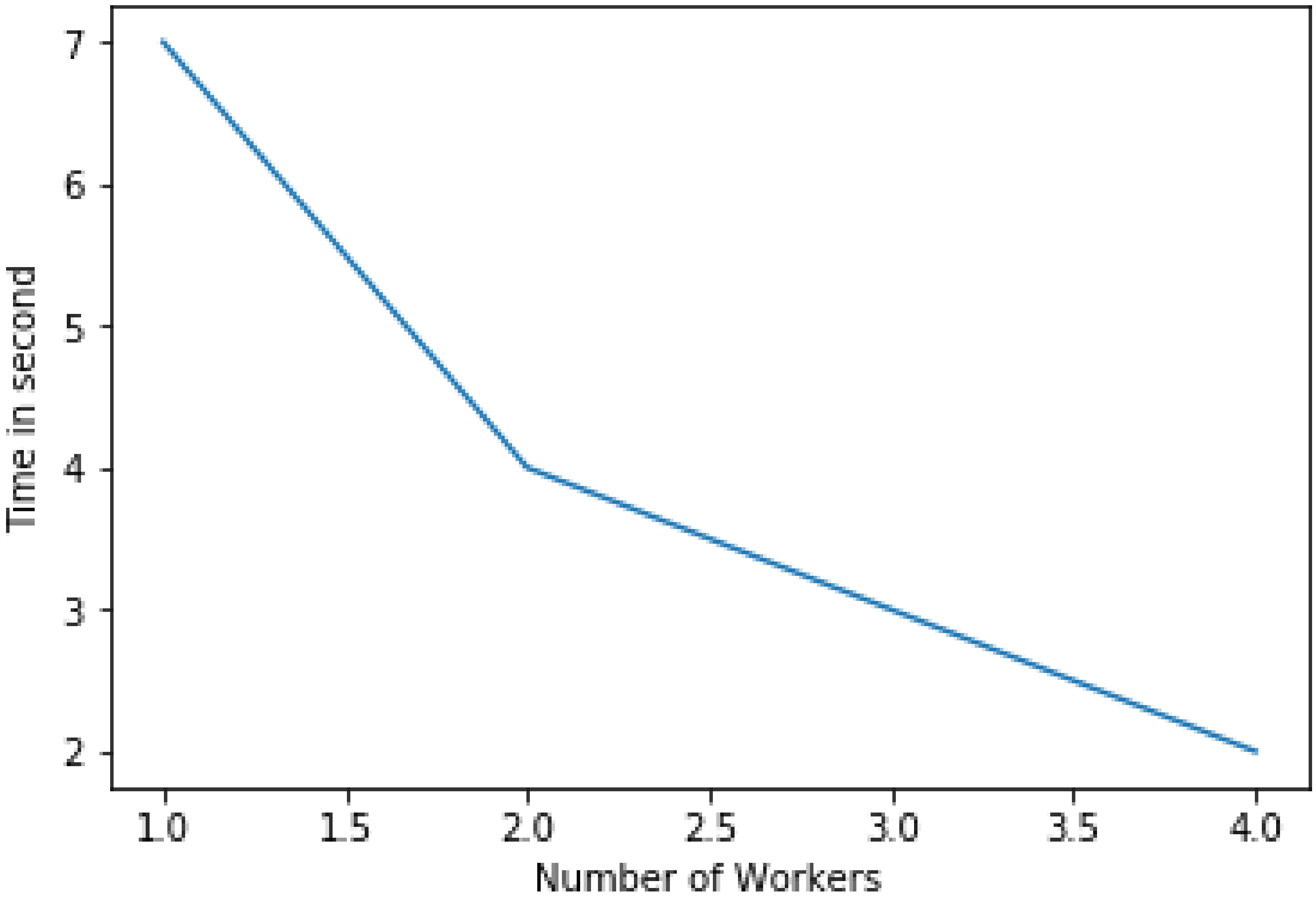
Structure of Code

- Main algorithm part is in lambda function.
- Initialization of TprocessExecutor and Map() function follows algorithm.
- New DataLoaderCopyMP() function is also implemented to use in Variable Importance.

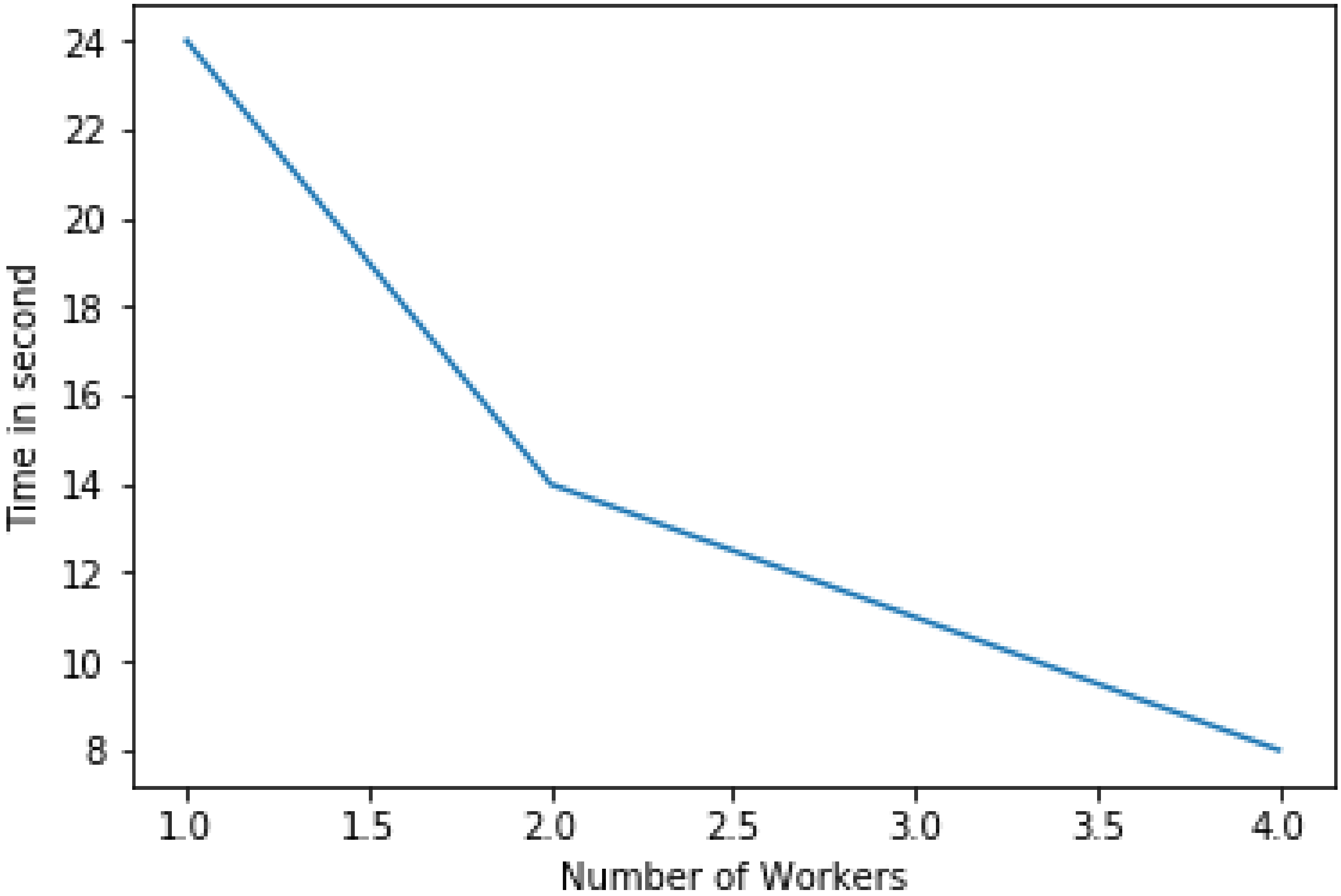


```
auto workItem = [&](UInt_t workerID) {

     //algorithm works here


    return value;
}

auto nWorkers = TMVA::gConfig().NWorkers();

ROOT::TProcessExecutor workers(nWorkers);
answer_vector = workers.Map(workItem, ROOT::TSeqI(numberOfIteration));
```

Graph of Performance change for CV



Graph of Perfomance Change for VI(kAll)


Time in second
Number of Workers