Integration of Spark parallelization in TMVA

Georgios Douzas

Supervisors: Enric Tejedor, Sergei Gleyzer

Spark engine

- A generalized framework for distributed data processing.
- Implemented in Scala.
- Provides a Python API called PySpark.
- Two main concepts:

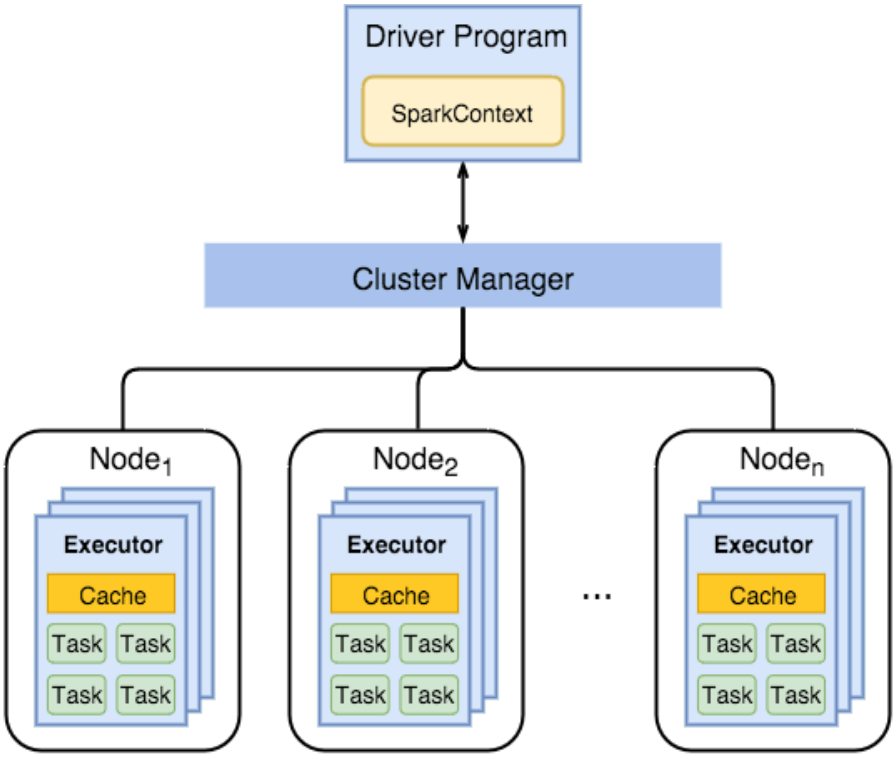
- RDD (Resilient Distributed Datasets)

- DAG (Direct Acyclic Graph)

Spark engine

- RDD is an immutable parallel data structure.
- DAG is a programming model for distributed systems.
- RDD operations: Transformations and Actions.

Spark architecture



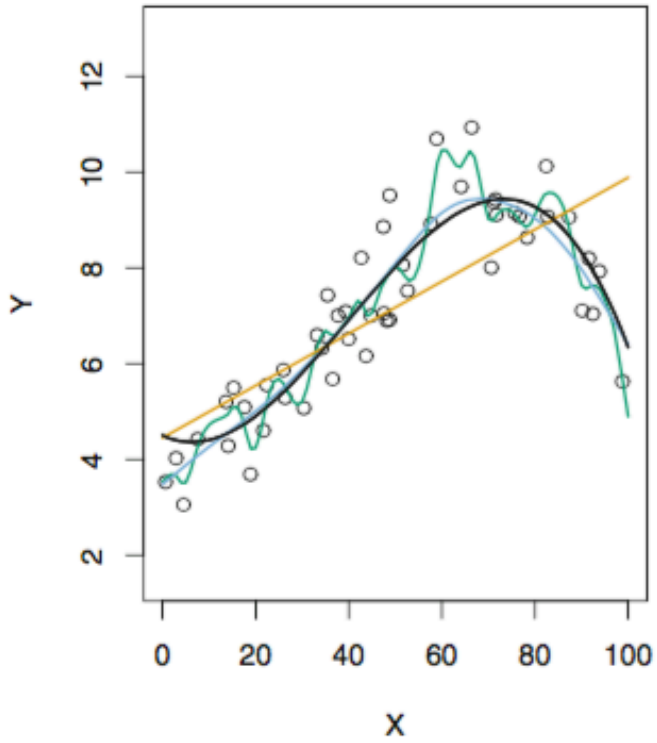
Parallelization of the TMVA code

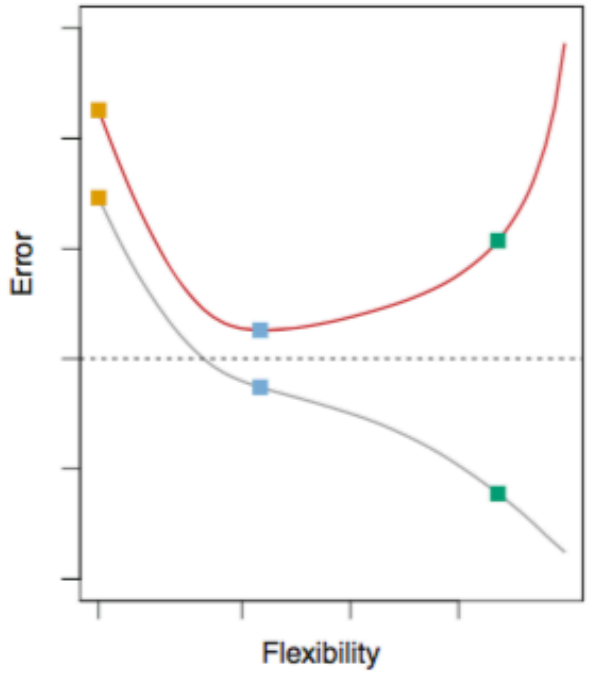
- Identify opportunities for parallelism.
- Examine whether the parallelism improves performance.
- Target on loops that include independent calculations.
- Use the same interface as the C++ TMVA code.

Parallelization in TMVA

- Cross validation.
- Optimization of tuning parameters.
- Local search for the optimization of tuning parameters.

Cross validation

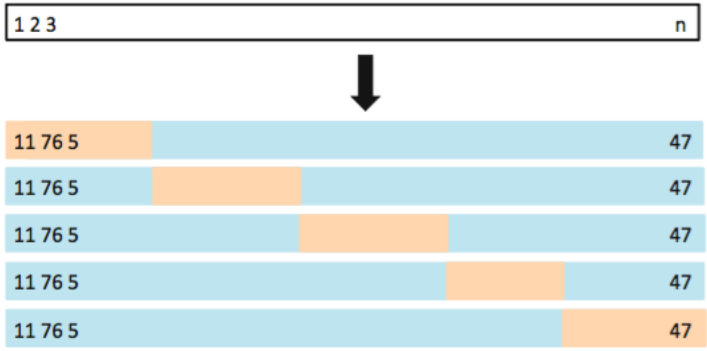




Cross validation

Validation

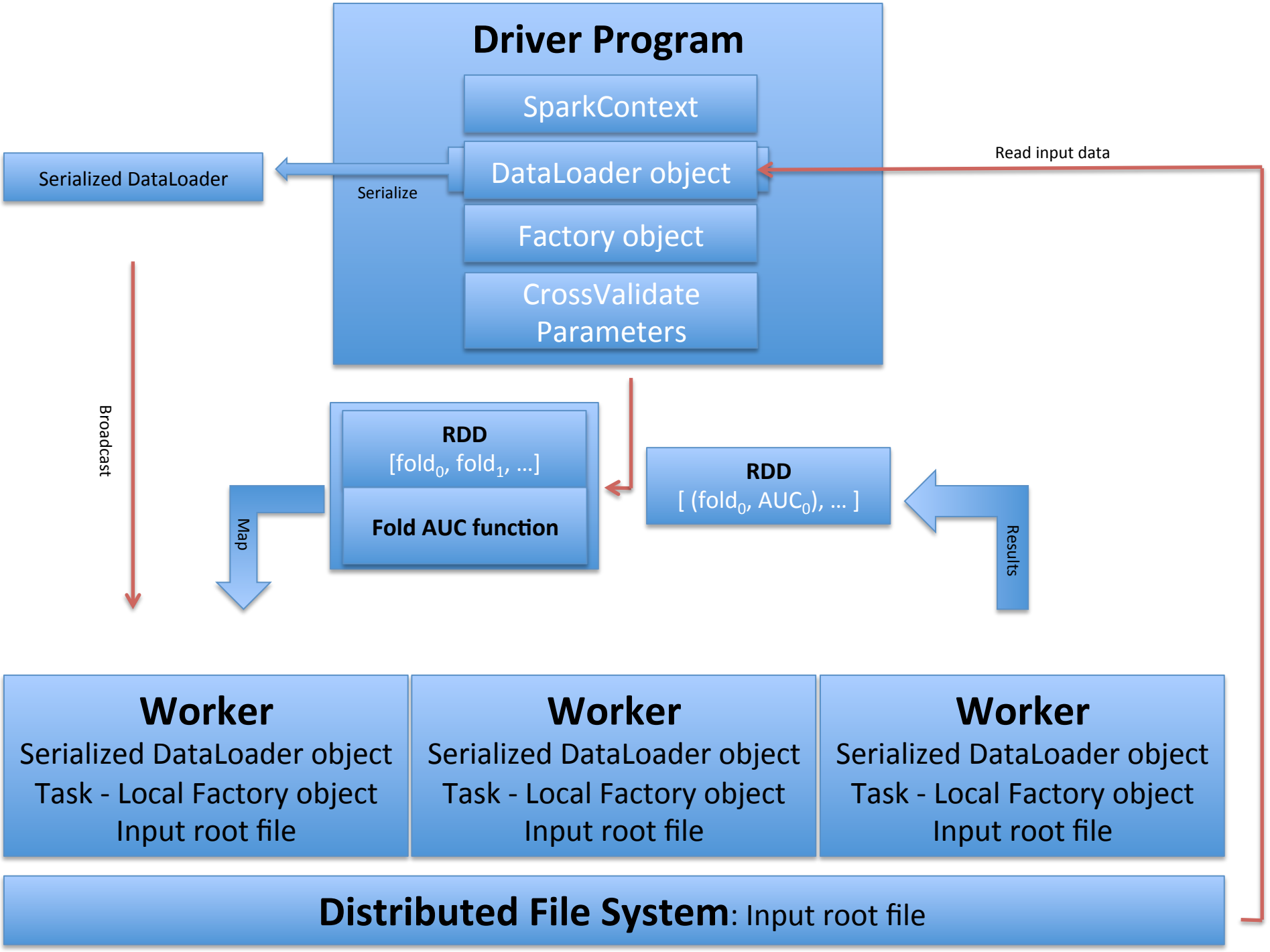
K-fold cross validation



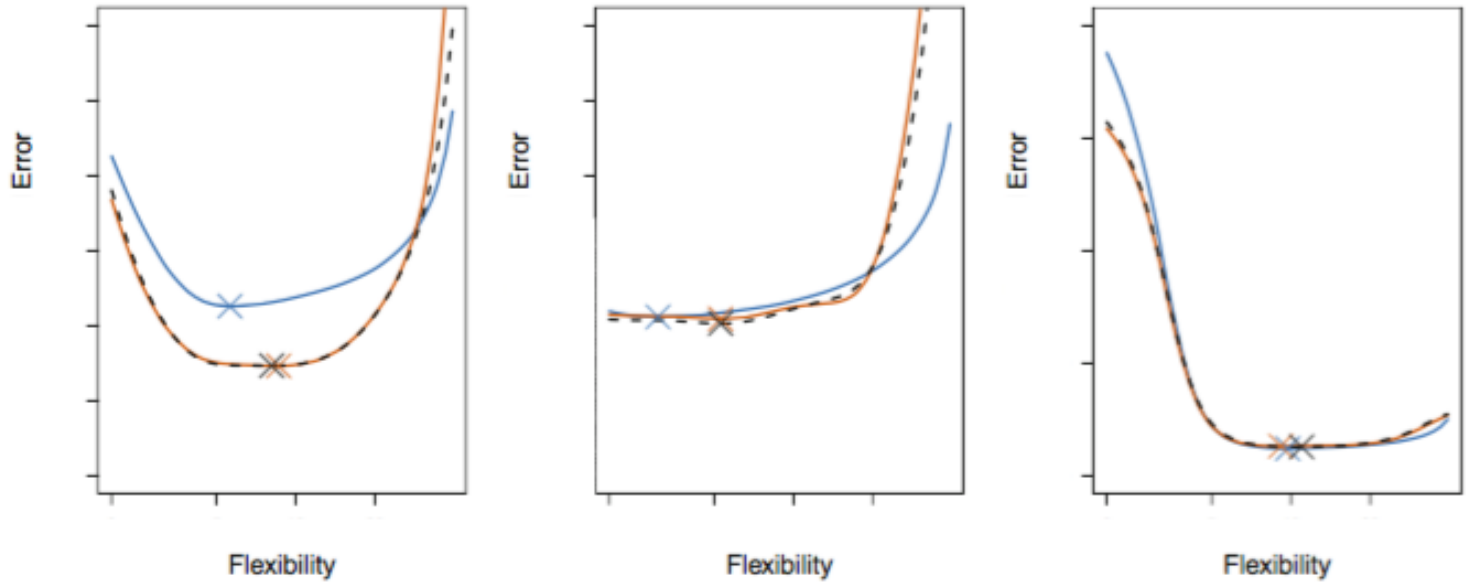
$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \mathrm{E}_i$$

Parallelized CrossValidate

- RDD = sc.parallelize([$fold_0$, $fold_1$, …, $fold_{k-1}$]).
- A map transformation is applied to the RDD.
- A new RDD with an AUC value for each fold index is returned.
- The average AUC is calculated.

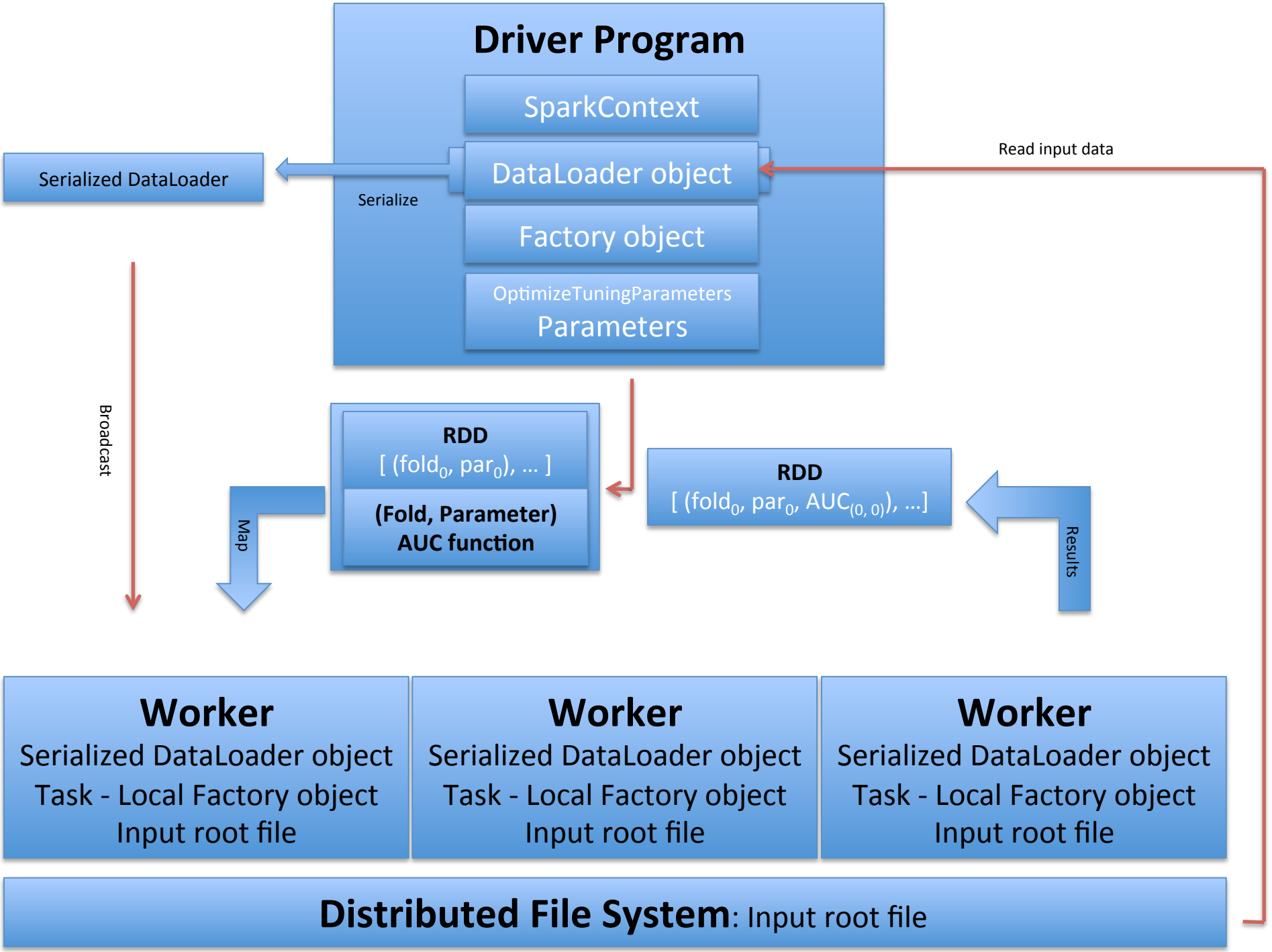

Driver Program
SparkContext
DataLoader object
Factory object
CrossValidate Parameters
Serialized DataLoader
Serialize
Read input data
Broadcast
RDD
[fold0, fold1, ...]
Fold AUC function
Map
RDD
[(fold0, AUC0), ...]
Results
Worker
Serialized DataLoader object
Task - Local Factory object
Input root file
Worker
Serialized DataLoader object
Task - Local Factory object
Input root file
Worker
Serialized DataLoader object
Task - Local Factory object
Input root file
Distributed File System: Input root file

Optimization of tuning parameters

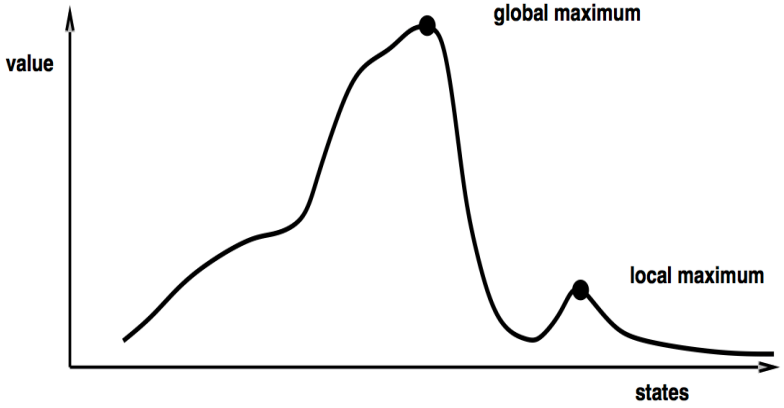


Parallelized OptimizeTuningParameters (Full search of parameter space)

- A default parameter space is defined.
- RDD = sc.parallelize([($fold_{0,}$ par_0), …, ($fold_{k-1,}$ par_0)…, ($fold_{0,}$ par_{p-1}), …, ($fold_{k-1,}$ par_{p-1})])
- A map transformation is applied to the RDD.
- A new RDD with an AUC value for each fold and parameter index is returned.
- The maximum AUC in each fold is calculated.
- The cross validation AUC is calculated for each "fold winner" parameter.


Driver Program
SparkContext
DataLoader object
Factory object
OptimizeTuningParameters
Parameters
Serialized DataLoader
Serialize
Read input data
Broadcast
RDD
[(fold0, par0), …]
(Fold, Parameter)
AUC function
Map
RDD
[(fold0, par0, AUC(0, 0)), …]
Results
Worker
Serialized DataLoader object
Task - Local Factory object
Input root file
Worker
Serialized DataLoader object
Task - Local Factory object
Input root file
Worker
Serialized DataLoader object
Task - Local Factory object
Input root file
Distributed File System: Input root file

Parallelized OptimizeTuningParameters (Local search of parameter space)

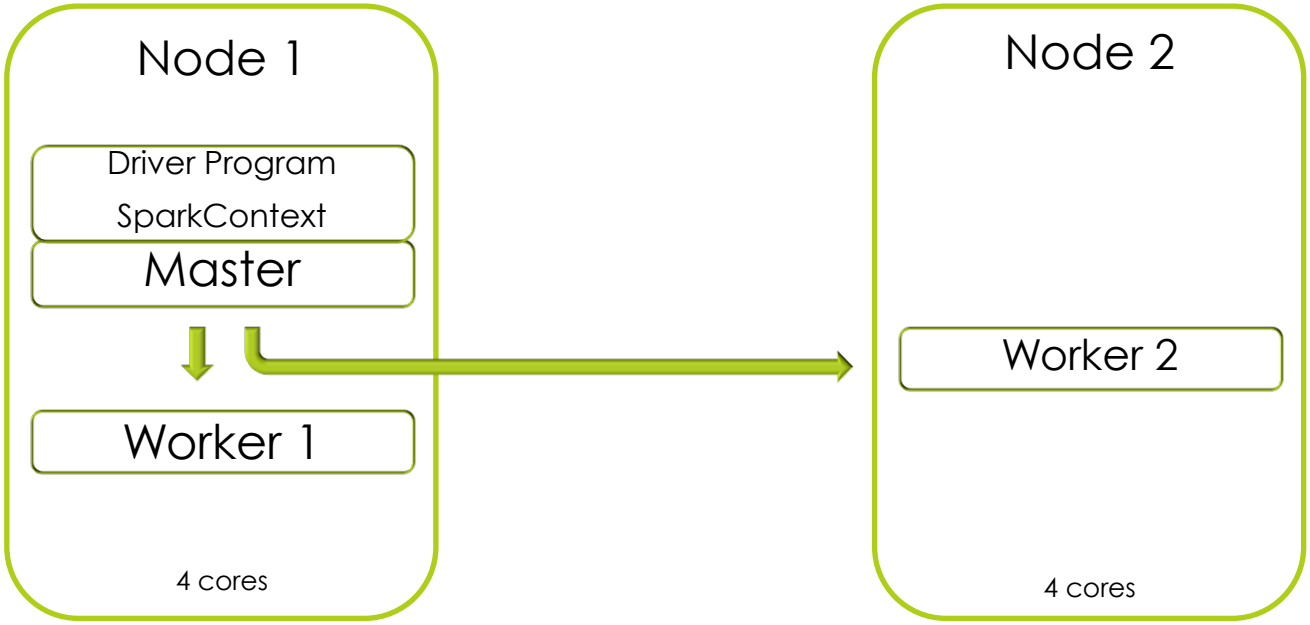


Hill-climbing search

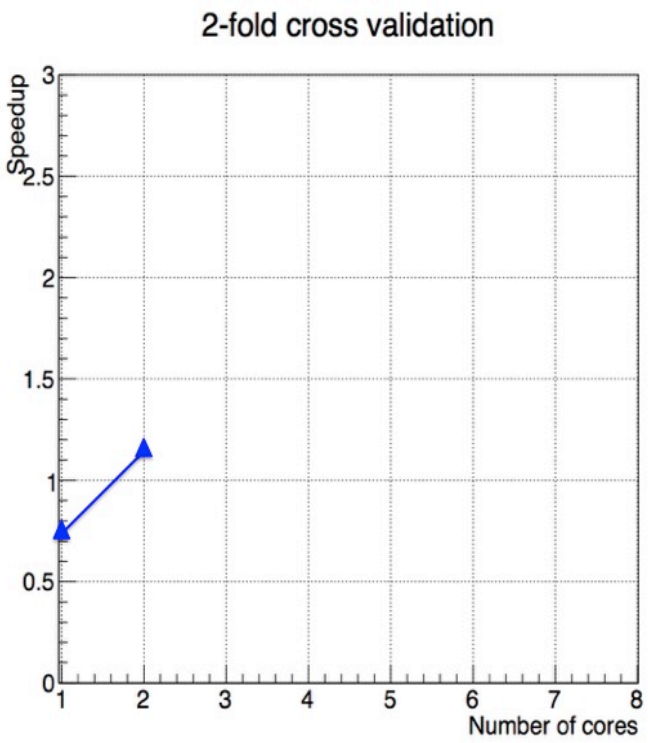
```
function HILL-CLIMBING(problem) return a state that is a local maximum
    input: problem, a problem
    local variables: current, a node.
                     neighbor, a node.

    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest valued successor of current
        if VALUE [neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
```

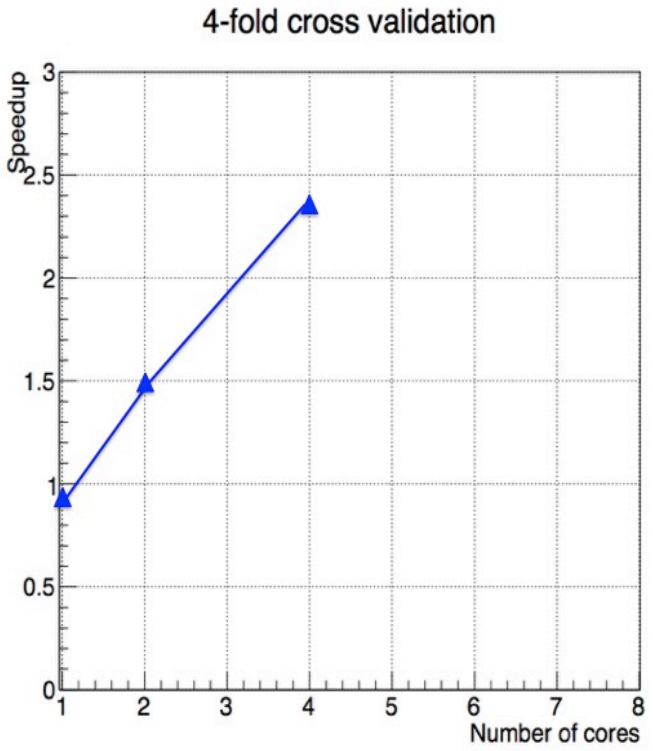
- For each fold a H.C. algorithm is applied.
- Parallelize any calculation in each H.C. iteration.
- RDD includes a subset of all the folds/parameters pairs.

Spark cluster

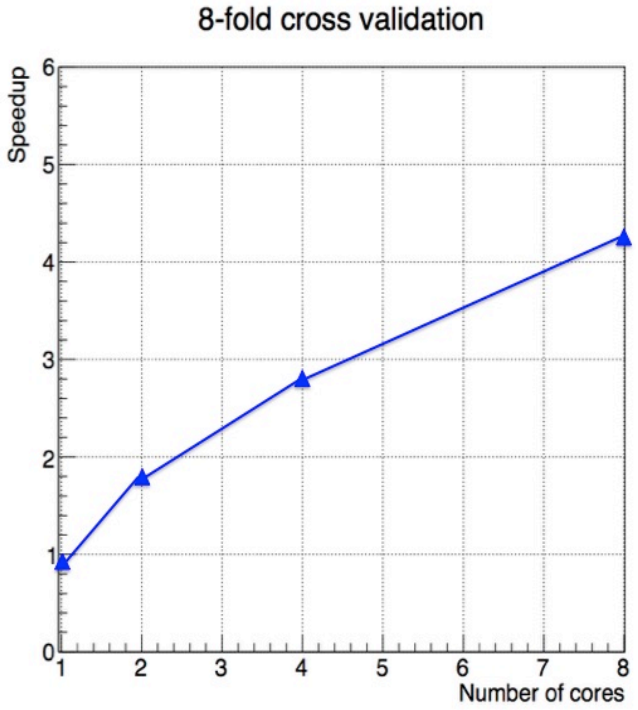


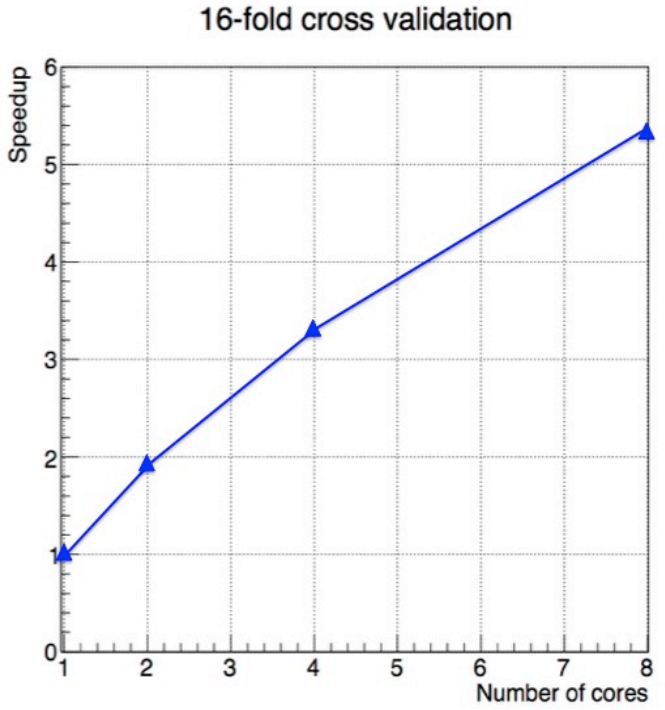
Experimental results



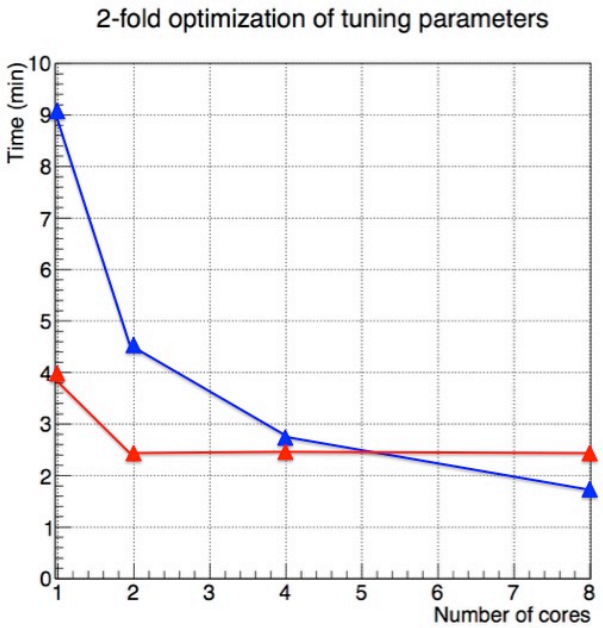


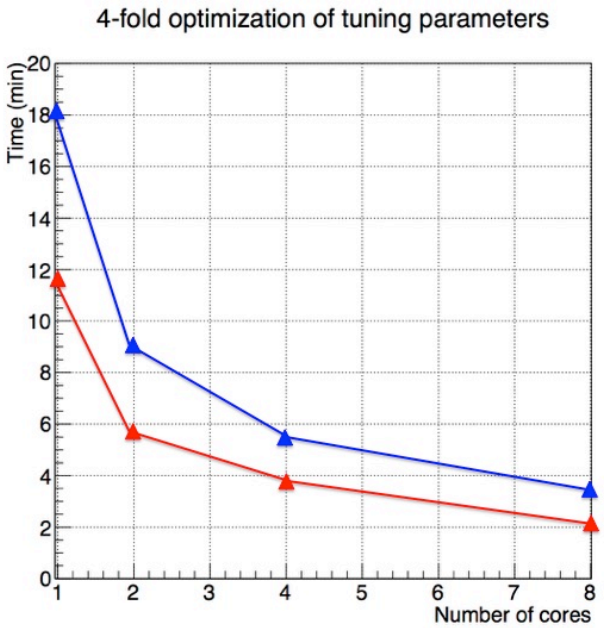
Experimental results





Experimental results





Full search

Hill climbing