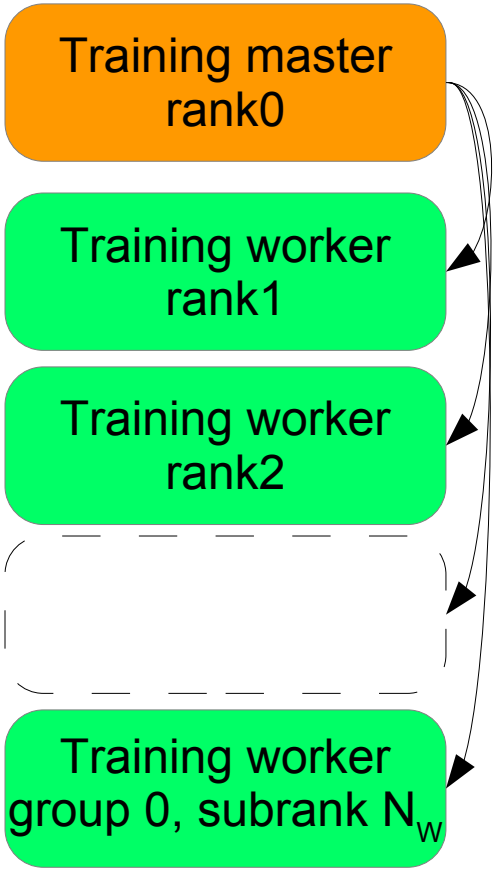
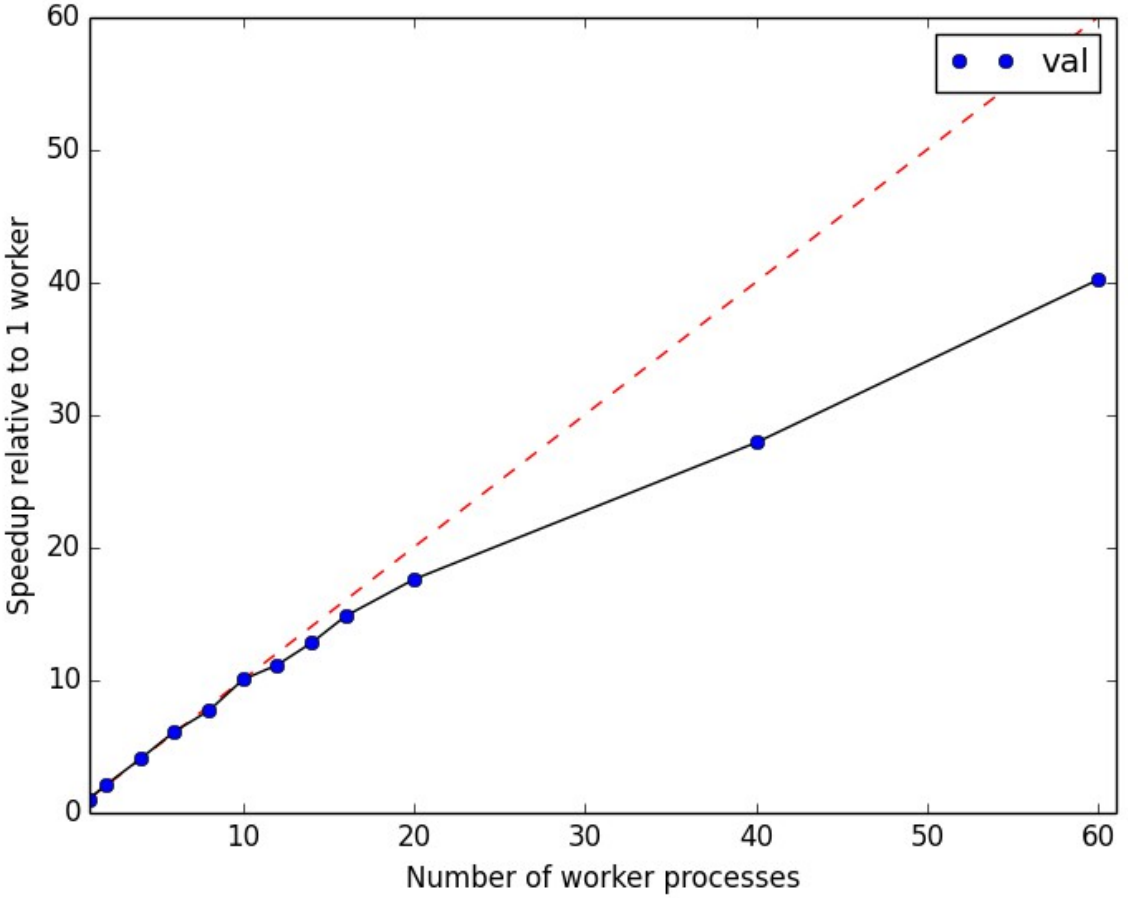
mpi-opt / mpi-learn

Distributed
training & optimization

mpi-learn

https://github.com/duanders/mpi_learn
https://arxiv.org/abs/1712.05878





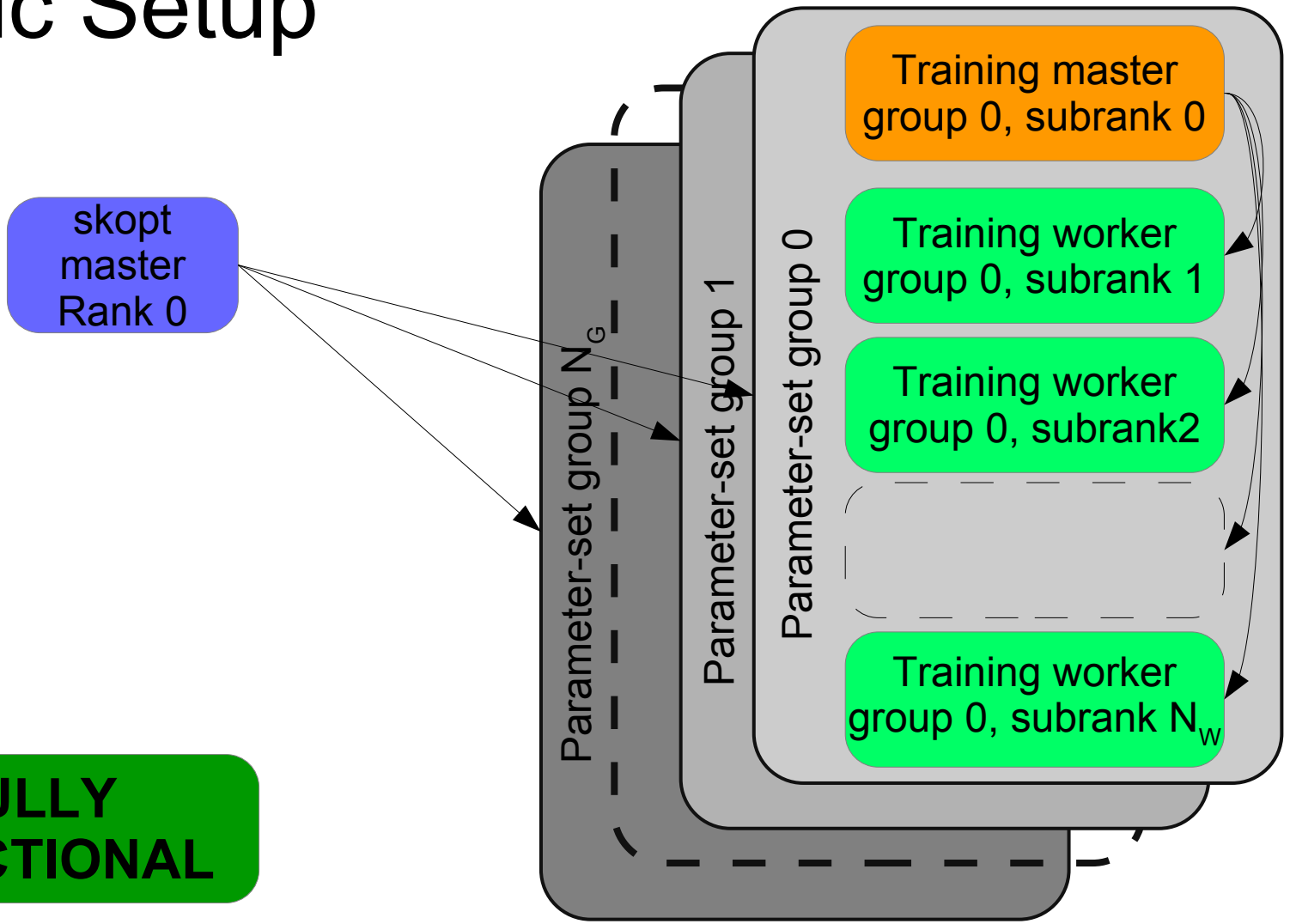
Goal

Provided a dataset, and provided a problem (type, input, output), obtain the optimum model for a given figure of merit (fom) in littel time, using efficiently as much resource as possible.

Input

- A dataset with input and target
- A problem type (regression, classification, ...)
- A function pset→model
- An HPC

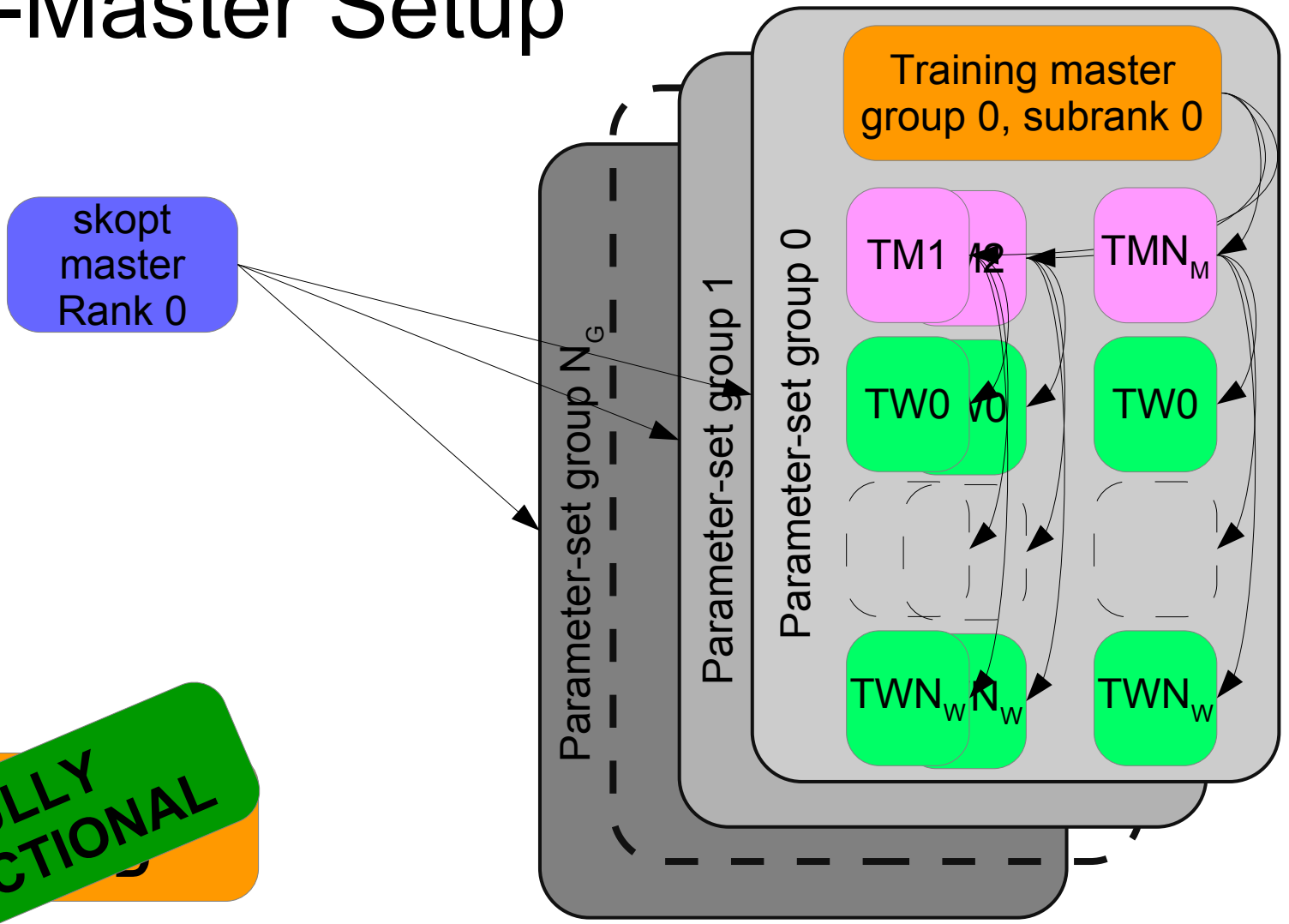
Basic Setup



FULLY FUNCTIONAL

- One master running the bayesian optimization
- N_G groups of nodes training on a parameter-set on simultaneously
  - One training master
  - N_W training workers

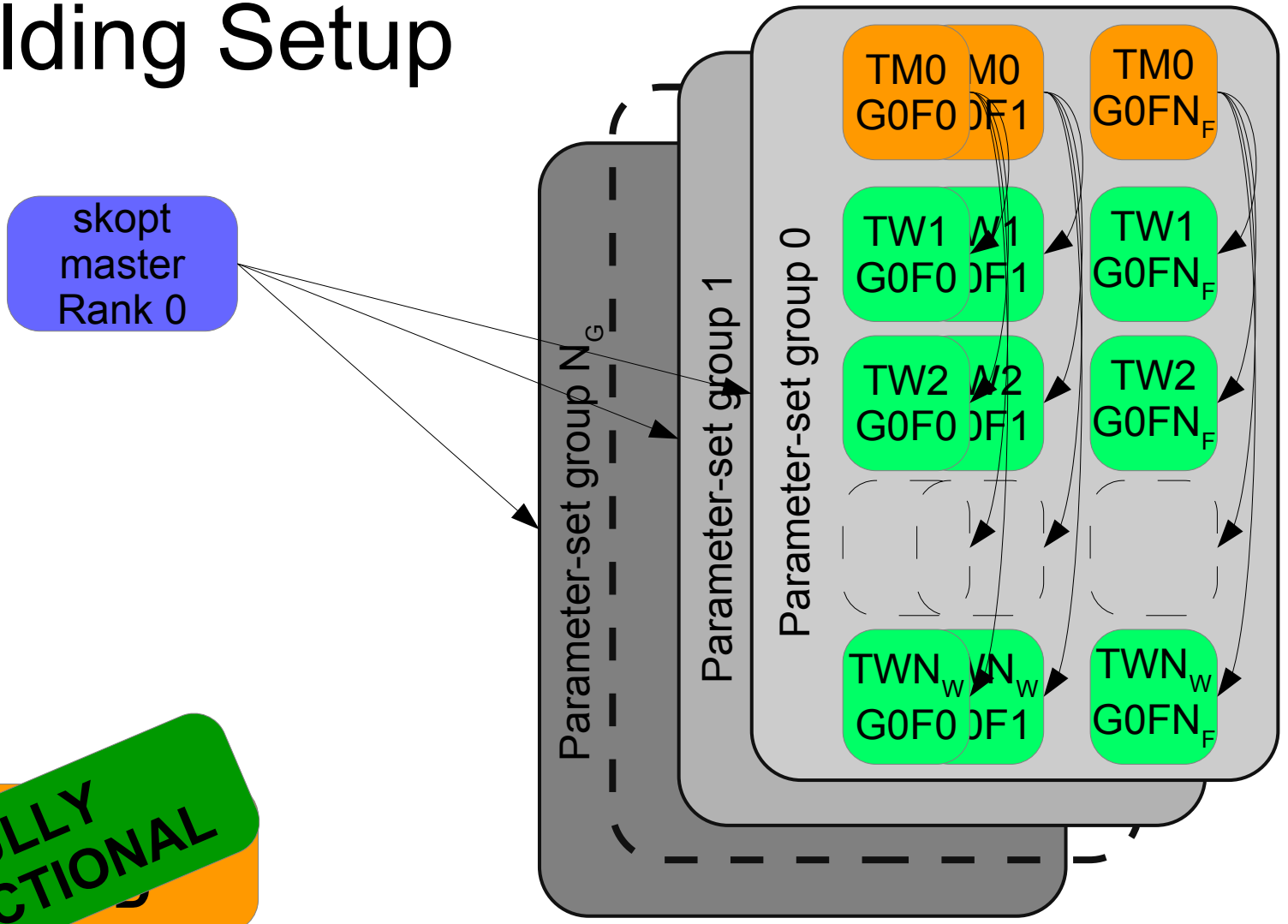
Sub-Master Setup



FULLY FUNCTIONAL

- One master running the bayesian optimization
- N_G groups of nodes training on a parameter-set on simultaneously
  - One training master
    - N_M training sub-masters
      - N_W training workers

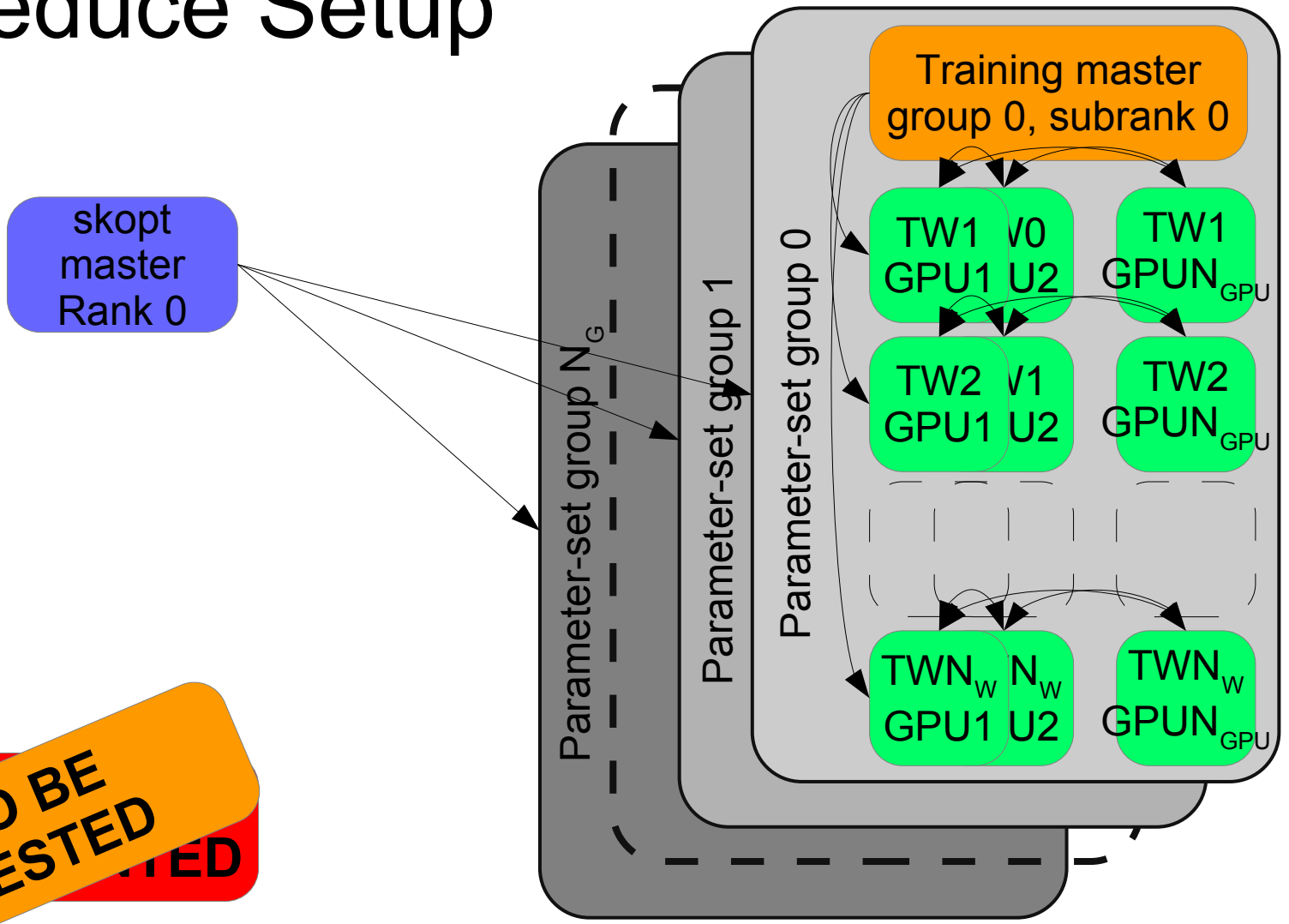
K-folding Setup



FULLY FUNCTIONAL

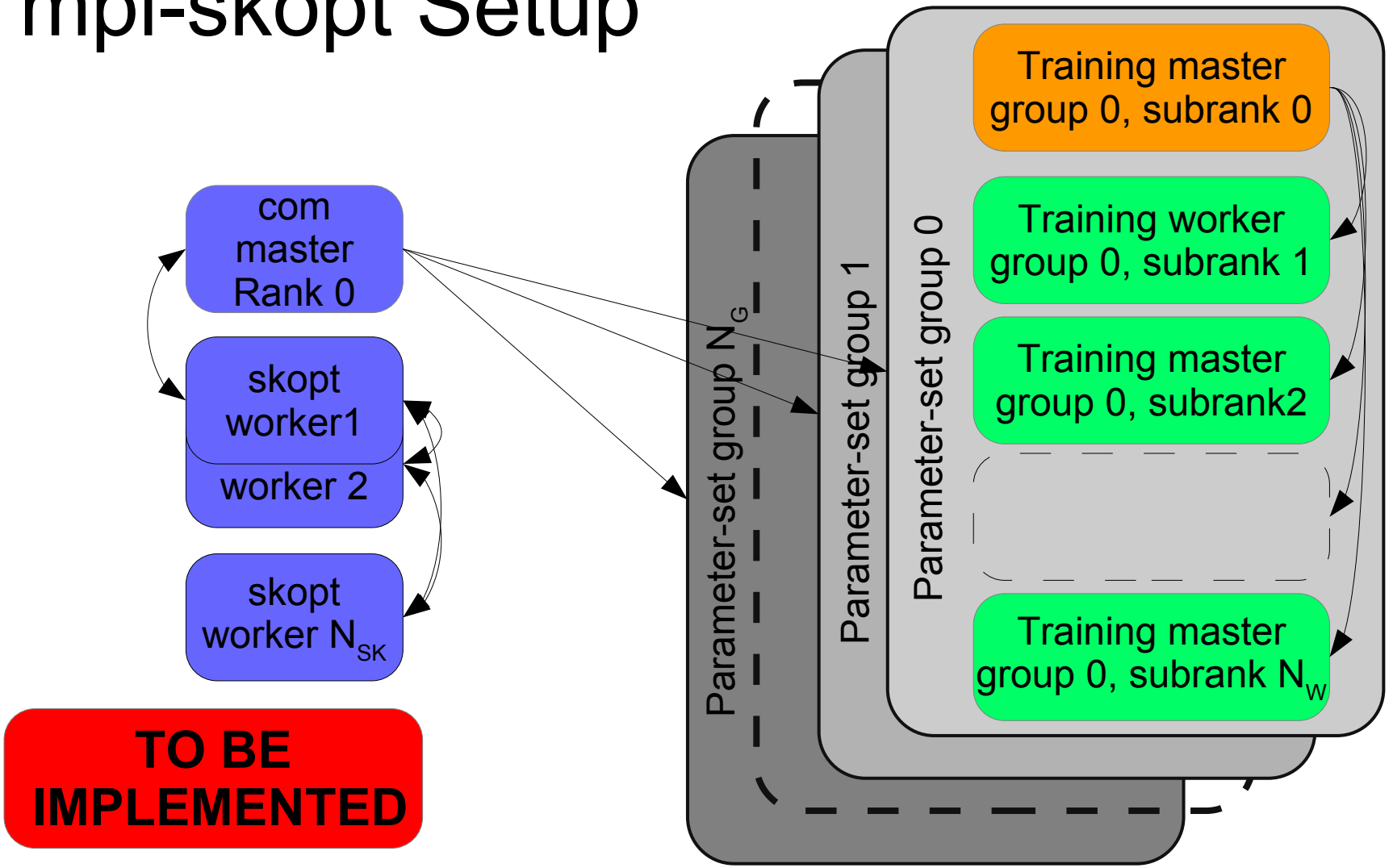
- One master running the bayesian optimization. Receiving the average *fom* over N_F folds of the data
- N_G groups of nodes training on a parameter-set on simultaneously
  - N_F groups nodes running one fold each
    - One training master
    - N_W training workers

all-reduce Setup



- One master running the bayesian optimization
- N_G groups of nodes training on a parameter-set on simultaneously
  - One training master
  - N_W training worker groups
    - N_{GPU} used for each worker group (either nodes or gpu)

mpi-skopt Setup



- One master running communication of parameter set
- N_{SK} workers running the bayesian optimization
- N_G groups of nodes training on a parameter-set on simultaneously
  - One training master
  - N_W training workers

Achievements

- Tested the **K-Folding** mechanism within mpi-opt
  - → Remains to take advantage of the additional and more accurate information with skopt. Need to do the math, and implement it
- → Tested **sub-master mechanism** within mpi-opt
  - ✓ Clarified when it is meant to be useful (downpour : not sound, easgd : reduces the idling of workers)
  - → Remains to profile and estimate advantage on a more complex problem than mnist, which converges too fast (we have candidate problems)
- Implemented structural changes for **multi-node workers**
  - → Remains to integrate horovod for distributed gradient computation
  - → Remains to tie loose ends on process communications inside a worker
- Implemented the **pytorch functionality**
  - ✓ Enables running on titan-ORNL
  - ✓ Enables multi-gpu gradient computation (summit)
  - → Remains to streamline dual keras-torch support in mpi-opt/learn
- Getting familiar with **mpi profiling** : tau
- Moving forward with mpi-cuda installation
- ✗ Not touch any of the skopt parallelisation

Next

Move on with scaling

- Tie loose ends on GAN figure of merit computation
  - ➔ Perform a large scale optimization
  - ➔ Identify bottlenecks
- Interface with a more complex problem, like TOPCLASS
  - ➔ Perform a large scale optimization
  - ➔ Identify bottlenecks

Further on scaling, performance and enabling

- Conclude torch integration (useful on nGPU/node HPC : summit/dev)
- Conclude horovod integration (useful on 1GPU/node HPC : CSCS)
- Conclude K-Folding (should help opt-convergence)