Lomonosov Moscow State University
Department of the Computational mathematics and cybernetics

Optimization of Neural networks training with Vector-Free heuristic on Apache Spark

K.G. Khamitov
N.N. Popova

GRID 2018,Dubna

Goals and topics

Research Goals

- Choose platform-independent heuristics which reduces amount of non-linear and memory-bound vector operations
- Determine limits of applicability of such memory-bound elimination heuristics for the classical SGD-based algorithms: Adam, AdaGrad, and Quasi-Newton based L-BFGS to make it applicable on the MapReduce platforms
- Determine the amount of efficiency improvements for such heuristics

Why it’s relevant?

- Map-Reduce based clusters like Apache Spark become so popular nowadays
- The necessity of adaptation current ML techniques which are used on other platforms to Apache Spark

Local Optimization methods for testing

Several popular ML methods are chosen for the computational experiment:

- L-BFGS(Quasi-Newton method)
- AdaGrad (Adaptive)
- Adam(Adaptive methods + momentum estimation)

Vector-Free optimizations

Vector-Free optimization[1] – replacing amount of memory-bound and non-linear vector operations with similar amount of scalar operations, which enables to reduce amount of Map and Reduce operations

Main approaches:

- Decomposing the vectors on some basis which depends only on a previous iterations.
- Replacing with manipulation on vectors directly to manipulations on such basis.
- Linearizing non-linear operations with power series decomposition.

[1] Weizhu Chen, Zhenghao Wang, and Jingren Zhou(2014) “Large-scale L-BFGS using mapreduce”. In Advances in Neural Information Processing Systems, p 1332–1340.

Example of Vector-Free optimization for L-BFGS method

Vector-free L-BFGS two-loop recursion

Input: $(2m+1)*(2m+1)$ dot product matrix between b_i
Output: The coefficients δ_i where $i = 1, 2, \ldots 2m+1$

1 **for** $i \leftarrow 1$ **to** $2m+1$ **do**
2 $\quad \delta_i = i \le 2m \,?\, 0 : -1$
3 **end**
4 **for** $i = k-1$ **to** $k-m$ **do**
5 $\quad j = i - (k-m) + 1$;
6 $\quad \alpha_i \leftarrow \frac{s_i \cdot p}{s_i \cdot y_i} = \frac{b_j \cdot p}{b_j \cdot b_{m+j}} = \frac{\sum_{l=1}^{2m+1} \delta_l b_l \cdot b_j}{b_j \cdot b_{m+j}}$;
7 $\quad \delta_{m+j} = \delta_{m+j} - \alpha_i$;
8 **end**
9 **for** $i \leftarrow 1$ **to** $2m+1$ **do**
10 $\quad \delta_i = (\frac{b_m \cdot b_{2m}}{b_{2m} \cdot b_{2m}})\delta_i$
11 **end**
12 **for** $i \leftarrow k-m$ **to** $k-1$ **do**
13 $\quad j = i - (k-m) + 1$;
14 $\quad \beta = \frac{b_{m+j} \cdot p}{b_j \cdot b_{m+j}} = \frac{\sum_{l=1}^{2m+1} \delta_l b_{m+j} \cdot b_l}{b_j \cdot b_{m+j}}$;
15 $\quad \delta_j = \delta_j + (\alpha_i - \beta)$
16 **end**

$10 < m < 16$

$$b_i = \begin{cases} x_{k-m+i} - x_{k-m+i-1} \; iff \; 0 \le i < m \\ \nabla f(x_{k-m+i}) - \nabla f(x_{k-m+i-1}) \; iff \; m \le i < 2m \\ f(x_k) \; iff \; i = 2m \end{cases}$$

2 Map operations on 4-6 and 12-16

L-BFGS two-loop recursion

Input: $\nabla f(x_k)$, s_i, y_i where $i = k-m, \ldots, k-1$
Output: new direction p

1 $p = -\nabla f(x_k)$;
2 **for** $i \leftarrow k-1$ **to** $k-m$ **do**
3 $\quad \alpha_i \leftarrow \frac{s_i \cdot p}{s_i \cdot y_i}$;
4 $\quad p = p - \alpha_i \cdot y_i$;
5 **end**
6 $p = (\frac{s_{k-1} \cdot y_{k-1}}{y_{k-1} \cdot y_{k-1}})p$
7 **for** $i \leftarrow k-m$ **to** $k-1$ **do**
8 $\quad \beta = \frac{y_i \cdot p}{s_i \cdot y_i}$;
9 $\quad p = p + (\alpha_i - \beta) \cdot s_i$;
10 **end**

AdaGrad

Pseudocode of AdaGrad

for j in 1 .. d

  for τ = 1..t

  $G(j,j) += (\nabla f_\tau(w))_j$

D=diagonalize(G)

for j in 1 .. d

  $w_j := w_j - \frac{\eta \nabla f_\tau(w)_j}{\sqrt{D_j}}$

Pseudo-code of VF-AdaGrad

$\nabla_w J(w_k) = \sum_0^m \delta_i r_i$

for i = 1, .. m

  $G(i,i) = \sum_0^m \delta_m r(m,i)$

for i = 1, ... m

  for j = 1... m

    $R(i,j) = (r_i, r_j)$

$\Delta w_k = -\eta(\sum_0^m \delta_m r_{ij}(r_i, e_j) + eI)^{-1/2} \nabla_w J(w_k)$

if $\sum_0^m \delta_m G(m,i) > min((r_i, e_j))$

  $remove\ r_0$

  $r_m := \nabla_w J(w_k)$

endif

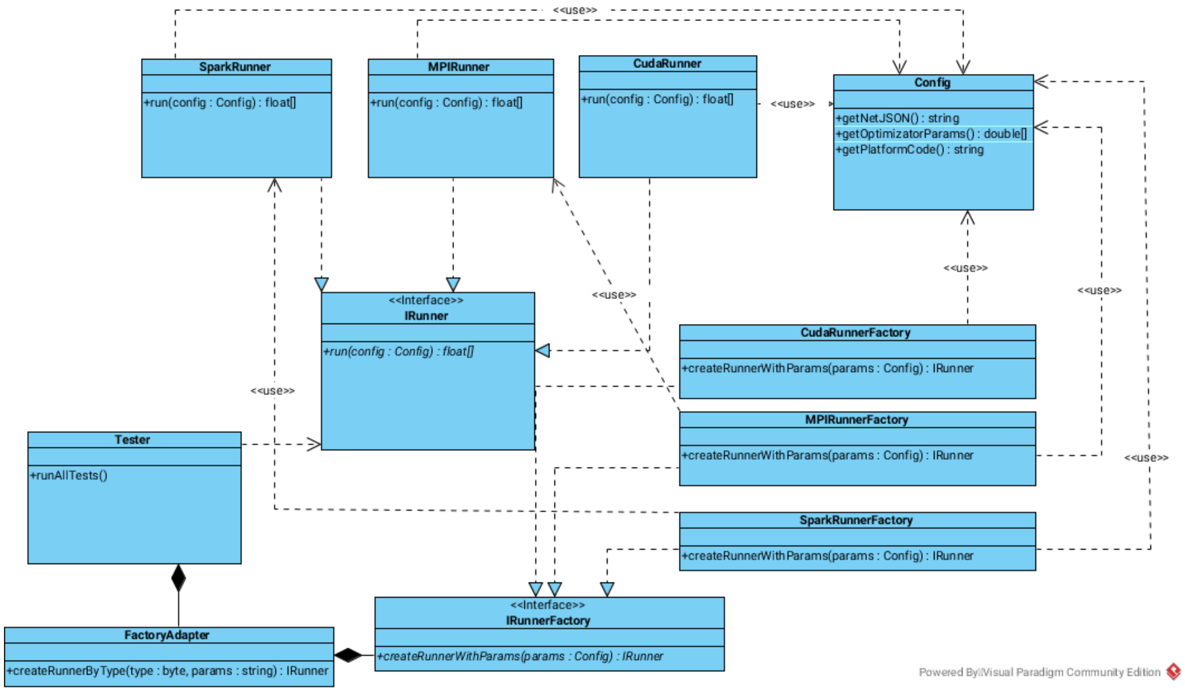
$recompute\ R(i,j)$

$w_{k+1} = w_k + \Delta w_k$

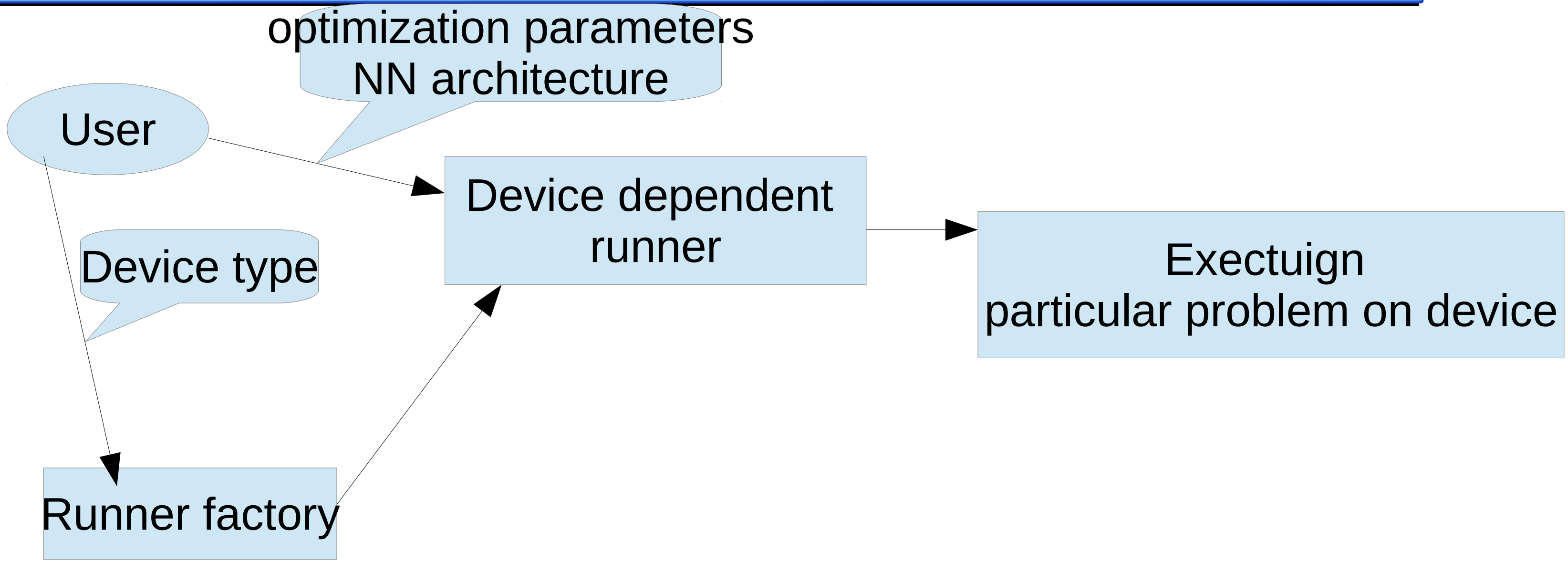
Used Packages & Architecture

- Spark 2.0.2 on Microsoft Azure 16 HD12v2 nodes
- SparkNet 0.1
- Custom wrapper to the TensorFlowTask from org.bytedeco.javacpp.tensorflow, which extends the same one from SparkNet
- Custom test system on python

Testing system architecture



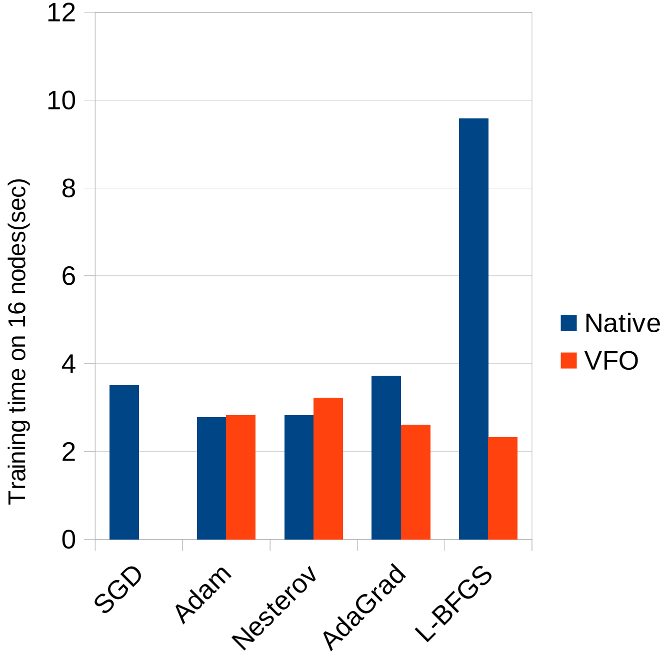
Testing sytsem architecture

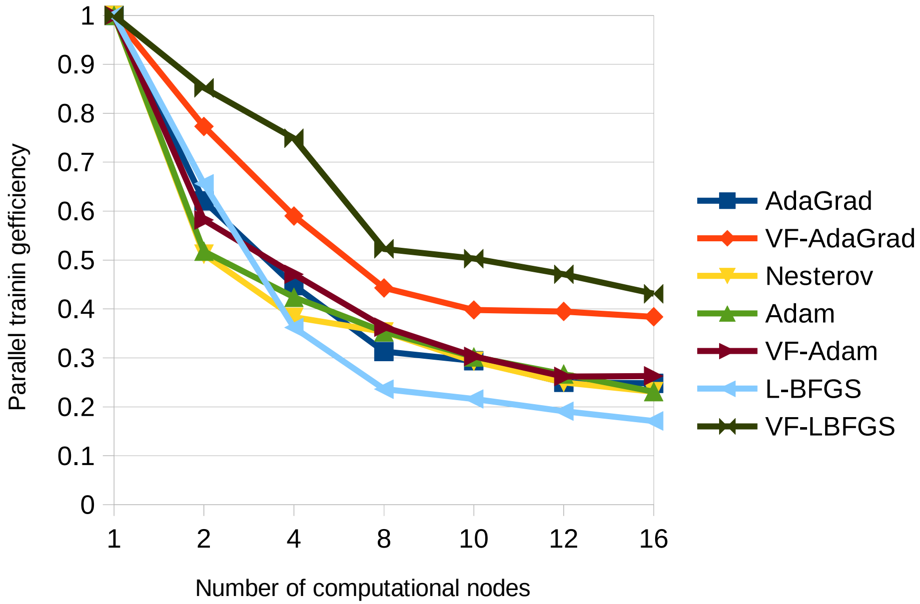


Experiments

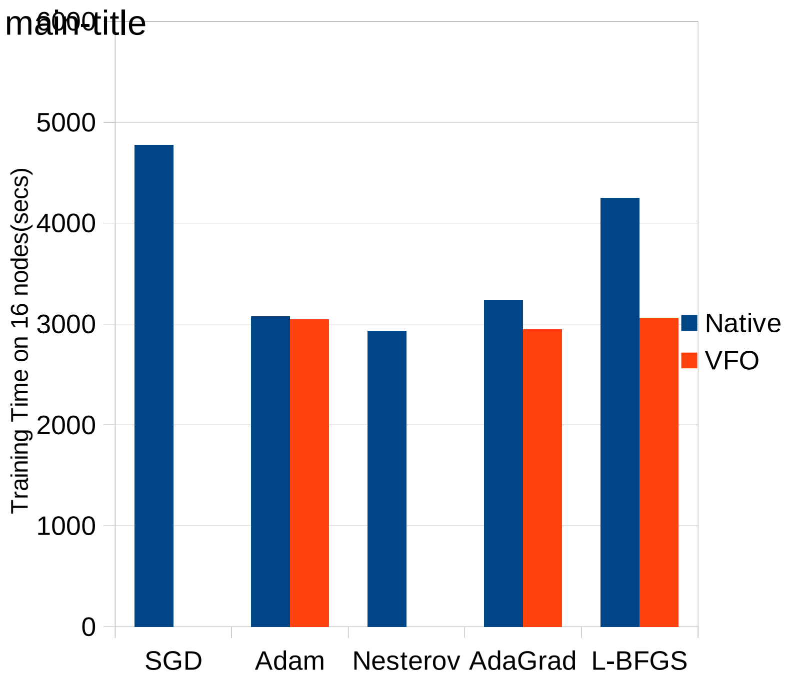
- NN architectures
  - MLP
  - VGG-16
  - LSTM
- Tasks
  - RNN – SILSO time series prediction db with LSTM
    - Batch size – 256 points
  - MLP – learning boolean functions
  - CNN – Image classifications on CIFAR-10 dataset
    - Batch size – 128 images

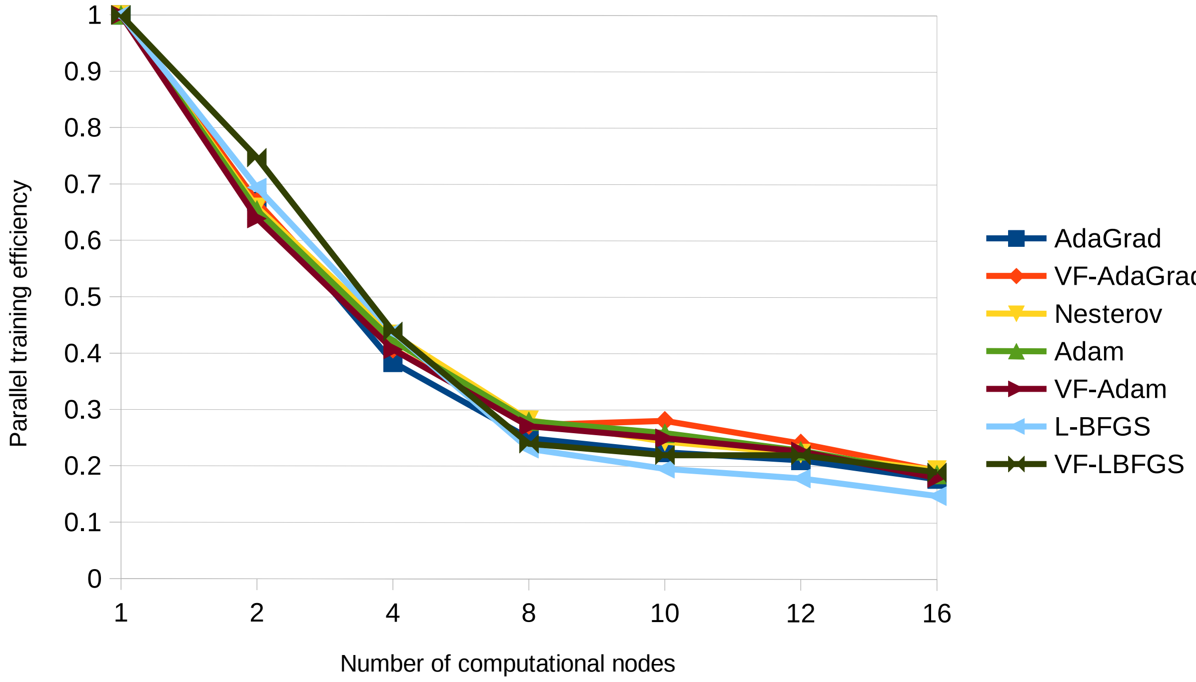
MLP training time comparison



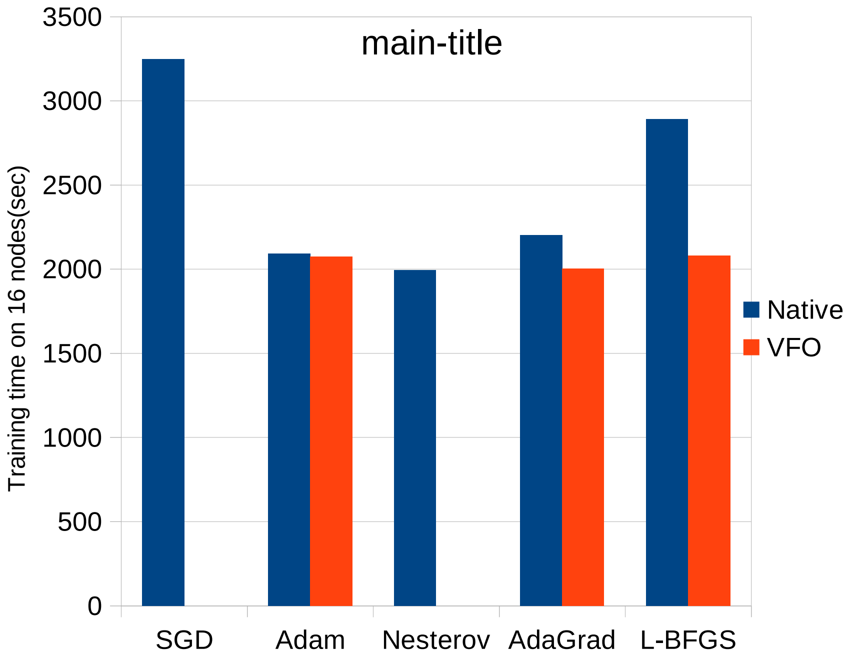


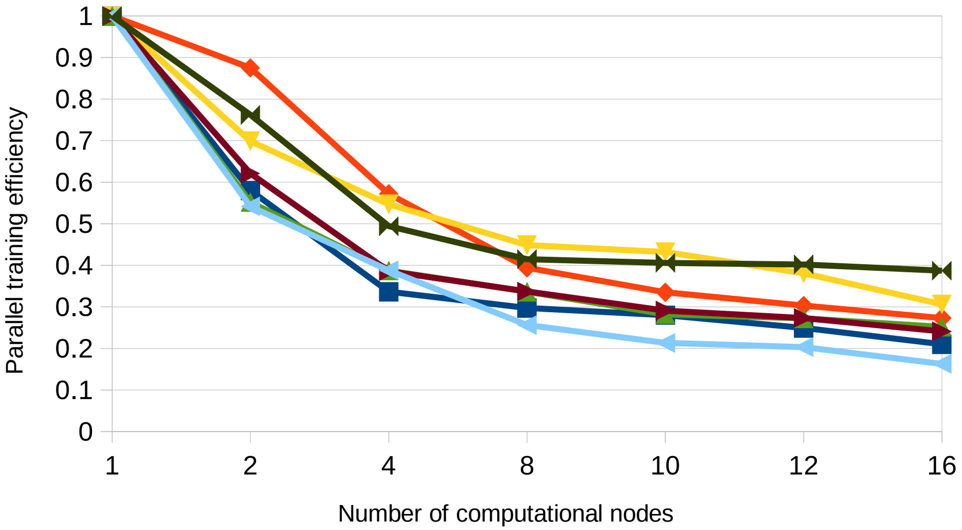
VGG-16 training time comparison on CIFAR-10





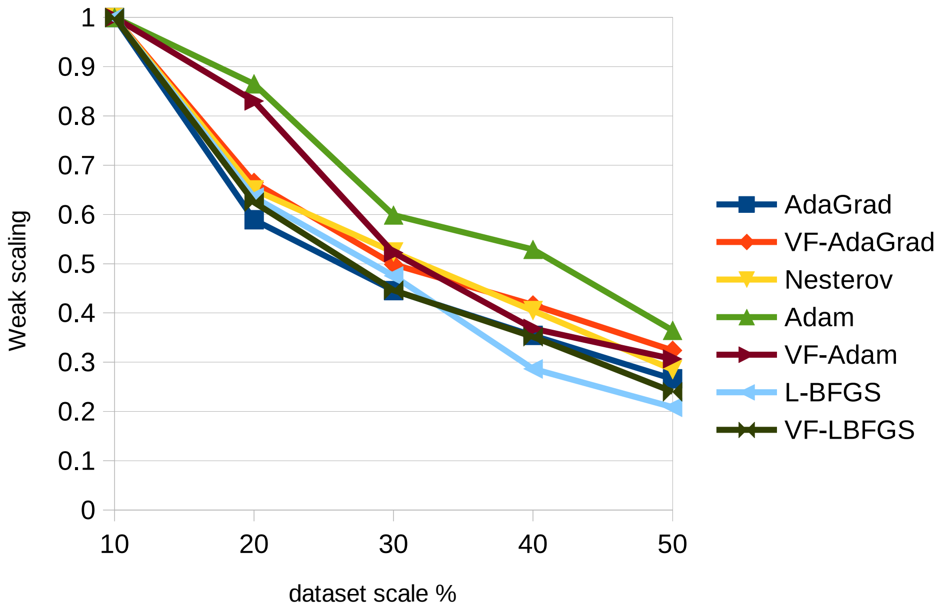
LSTM training time comparison on SILSO prediction task





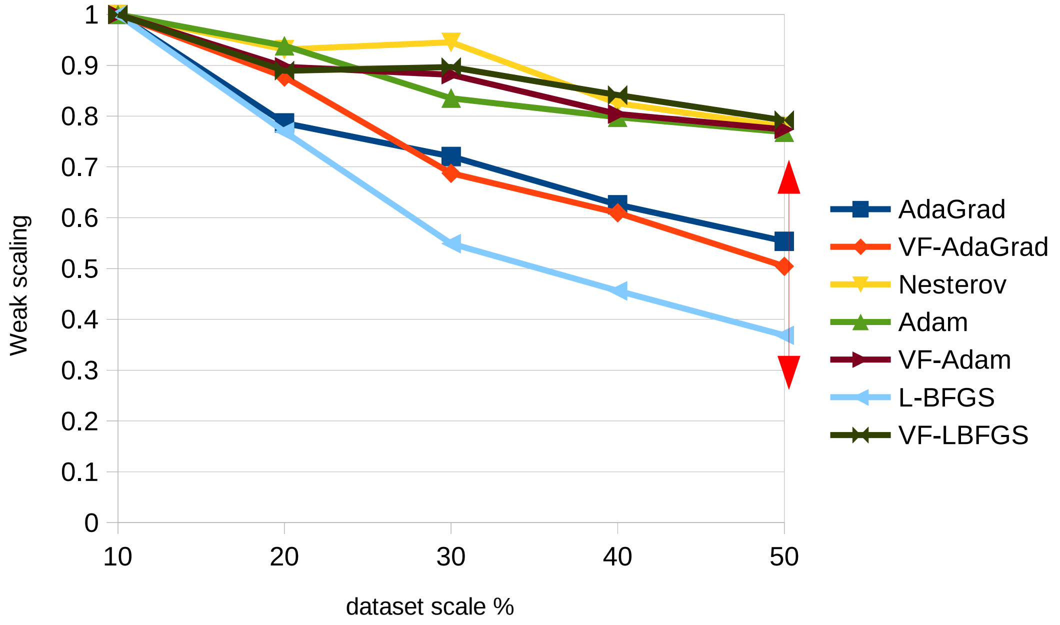
Weak scaling of LSTM Training

$$W_s(p) = \frac{TTS_{fxd}}{TTS_{fxd*p}}$$



Weak scaling of the VGG-16 training

$$W_s(p)=\frac{TTS_{fxd}}{TTS_{fxd*p}}$$



Summary

- The VF heuristic was chosen as platform-independent optimizing heuristic
- Popular optimizers for NN training like(L-BFGS, Adam, AdaGrad) were implemented with VF and non-VF versions for the experiments
- Testing platform for comparison such implementations of NN optimizers were implemented
- Results of the experiments shows that:
  - VF heuristic hardly applicable to the methods that contain a lot of linear vector operations(Adam/Nesterov momentum)
  - Applicability of heuristic depends on amount of non-linear operations in the main loop of the algorithm
  - Effects of such heuristic raises with the task complexity, the most improvements are shown on the LSTM training problem(x3.85)