A performance comparison of Deep Learning frameworks on KNL

R. Zanella, G. Fiameni, M. Rorro
Middleware, Data Management - SCAI - CINECA

IXPUG Bologna, March 5, 2018

Table of Contents

1. Problem description
2. Tested Neural Networks
3. Architectures and Software
4. Single node benchmark
5. Multinode benchmark
6. Conclusions

Neural Network Internals: classification problem

- Given a single input, a trained neural network is able to predict a distribution of probability:

$$\underbrace{\hat{\boldsymbol{y}}}_{\text{prediction}} = f(\underbrace{\boldsymbol{x}}_{\text{input}};\underbrace{\boldsymbol{\theta}}_{\text{NN parameters}})$$

Neural Network Internals: classification problem

- Given a single input, a trained neural network is able to predict a distribution of probability:

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x};\boldsymbol{\theta})$$

- NN paramaters are chosen in order to minimize the average error on a given training set $\left\{\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)}\right\}_{i=1,\dots,N}$:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{1}^{N} L\left(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),\boldsymbol{y}^{(i)}\right)$$

Neural Network Internals: classification problem

- Given a single input, a trained neural network is able to predict a distribution of probability:

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x};\boldsymbol{\theta})$$

- NN paramaters are chosen in order to minimize the average error on a given training set $\left\{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right\}_{i=1,\dots,N}$:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{1}^{N} L\left(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), \boldsymbol{y}^{(i)}\right)$$

- a Stochastic Gradient Descent (SGD) algorithm step is:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \epsilon_k \hat{\boldsymbol{g}}^{(k)} \qquad \text{where} \qquad \hat{\boldsymbol{g}}^{(k)} = \frac{1}{n}\nabla_{\boldsymbol{\theta}} \sum_{l\in\text{batch}} L(f(\boldsymbol{x}^{(l)};\boldsymbol{\theta}^{(k)}), \boldsymbol{y}^{(l)}), \quad n = \#\text{batch}$$

ImageNet Large Scale Visual Recognition Competition (ILSVRC):

- annual software contest (since 2010);
- tasks: image classification, object localization/detection, scene detection;

Tested Networks:

- AlexNet: winner of 2012 classification task,
- Overfeat: winner of 2013 localization task, remarkable results also in classification and detection
- VGG: winner of 2014 localization task, second place on classification

Russakovsky et al., ***ImageNet Large Scale Visual Recognition Challenge*****, 2015**

AlexNet (Krizhevsky, Sutskever, Hinton, 2012)

Structure:

- five convolutional layers:
  - kernel sizes 11×11, 5×5, three 3×3;
  - output channels 64, 192, 384, 256, 256;
- three max-pool layers (kernel 3×3, stride 2×2);
- three fully-connected layers;
- Rectified Linear Unit (ReLU) as nonlinear activation function.

Krizhevsky, Sutskever, Hinton, ***ImageNet Classification with Deep Convolutional Neural Networks*****, 2012**

AlexNet (Krizhevsky, Sutskever, Hinton, 2012)

Structure:

- five convolutional layers:
  - kernel sizes 11×11, 5×5, three 3×3;
  - output channels 64, 192, 384, 256, 256;
- three max-pool layers (kernel 3×3, stride 2×2);
- three fully-connected layers;
- Rectified Linear Unit (ReLU) as nonlinear activation function.

A note on benchmarked network:

- original version exploited multiple GPUs, to overcome memory limits;
- benchmarked version: AlexNet_v2 (one device),

Krizhevsky, Sutskever, Hinton, ***ImageNet Classification with Deep Convolutional Neural Networks*****, 2012**

OverFeat (Sermanet et al., 2014)

Structure (fast model):

- five convolutional layers
  - kernel sizes 11×11, 5×5, three 3×3;
  - output channels 96, 256, 512, 1024, 1024;
- three max-pool layers (kernel 2×2, stride 2×2);
- three fully-connected layers;
- Rectified Linear Unit (ReLU) as nonlinear activation function

Sermanet et al., ***Overfeat: Integrated recognition, localization and detection using convolutional networks*****, 2014**

OverFeat (Sermanet et al., 2014)

Structure (fast model):

- five convolutional layers
  - kernel sizes 11×11, 5×5, three 3×3;
  - output channels 96, 256, 512, 1024, 1024;
- three max-pool layers (kernel 2×2, stride 2×2);
- three fully-connected layers;
- Rectified Linear Unit (ReLU) as nonlinear activation function

Differences w.r.t. AlexNet:

- remarkable size increase of output channels of convolutional layers (larger number of filters);
- no overlap on max-pool layers;

Sermanet et al., ***Overfeat: Integrated recognition, localization and detection using convolutional networks*****, 2014**

VGG (Simonyan, Zisserman, 2014)

Structure (vgg_a/vgg11 network):

- eight convolutional layers
  - fixed kernel size 3×3;
  - output channels 64, 128, 2×256, 4×512;
- five max-pool layers (kernel 2×2, stride 2×2);
- three fully-connected layers;
- Rectified Linear Unit (ReLU) as nonlinear activation function

Simonyan, Zisserman, ***Very Deep Convolutional Networks for Large-Scale Image Recognition*****, 2014**

VGG (Simonyan, Zisserman, 2014)

Structure (vgg_a/vgg11 network):

- eight convolutional layers
  - fixed kernel size 3×3;
  - output channels 64, 128, 2×256, 4×512;
- five max-pool layers (kernel 2×2, stride 2×2);
- three fully-connected layers;
- Rectified Linear Unit (ReLU) as nonlinear activation function

Differences w.r.t. AlexNet:

- slight size increase of output channels of convolutional layers;
- remarkable depth increase (conv layers: five $\rightarrow$ eight);
- smaller kernels;
- no overlap on max-pool layers;

Simonyan, Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014

CPU architectures:

- 2x Intel Broadwell, 2x Intel Xeon E5-2697 v4@2.3GHz, 36 cores (total), 128 GB RAM
- Intel Knights Landing, Intel Xeon Phi7250 @1.4GHz, 68 cores, 96 GB RAM (+16 GB MCDRAM)
- 2x Intel Skylake, 2x Intel Xeon 8160 @2.1GHz, 2x Intel Xeon 8160 @2.1GHz, 48 cores (total), 192 GB RAM

1

CPU architectures:

- 2x Intel Broadwell, 2x Intel Xeon E5-2697 v4@2.3GHz, 36 cores (total), 128 GB RAM
- Intel Knights Landing, Intel Xeon Phi7250 @1.4GHz, 68 cores, 96 GB RAM (+16 GB MCDRAM)
- 2x Intel Skylake, 2x Intel Xeon 8160 @2.1GHz, 2x Intel Xeon 8160 @2.1GHz, 48 cores (total), 192 GB RAM

GPU architectures[1]:

- Nvidia K80: 5.6 Tflops peak performance (sp), 24 GB RAM
- Nvidia P100 (PCIe): 10.6 Tflops peak performance (sp), 16 GB RAM

[1] **hosted on: Intel Haswell, 2x Intel Xeon 2630 v3 @2.4GHz, 16 cores (total), 128 GB RAM**

CPU architectures:

- 2x Intel Broadwell, 2x Intel Xeon E5-2697 v4@2.3GHz, 36 cores (total), 128 GB RAM
- Intel Knights Landing, Intel Xeon Phi7250 @1.4GHz, 68 cores, 96 GB RAM (+16 GB MCDRAM)
- 2x Intel Skylake, 2x Intel Xeon 8160 @2.1GHz, 2x Intel Xeon 8160 @2.1GHz, 48 cores (total), 192 GB RAM

GPU architectures[1]:

- Nvidia K80: 5.6 Tflops peak performance (sp), 24 GB RAM
- Nvidia P100 (PCIe): 10.6 Tflops peak performance (sp), 16 GB RAM

Knights Landing settings:

- Hyper-Threading is enabled;
- Memory Mode is cache;
- Cluster Mode is quadrant;
- Network type: Intel Omnipath, 100 Gb/s

[1] **hosted on: Intel Haswell, 2x Intel Xeon 2630 v3 @2.4GHz, 16 cores (total), 128 GB RAM**

Caffe

theano

TensorFlow

Caffe

- Berkeley Vision and Learning Center (BVLC)
- C++, Matlab, Python APIs
- prototxt text file for network definition
- GPU: CUDA, cuDNN, (+nccl for single-node multi-gpu)
- CPU: BLAS implementation (ATLAS, MKL, or OpenBlas)
- tested configuration: BVLC/Caffe 1.0.0, CUDA 8.0, cuDNN 6.0

- Berkeley Vision and Learning Center (BVLC)
- C++, Matlab, Python APIs
- prototxt text file for network definition
- GPU: CUDA, cuDNN, (+nccl for single-node multi-gpu)
- CPU: BLAS implementation (ATLAS, MKL, or OpenBlas)
- tested configuration: BVLC/Caffe 1.0.0, CUDA 8.0, cuDNN 6.0

- Intel branch of original project
- prerequisites: mkl-dnn (based on mklml)
- tested configuration: intel/Caffe 1.0.0, mkl-dnn 0.9

- Berkeley Vision and Learning Center (BVLC)
- C++, Matlab, Python APIs
- prototxt text file for network definition
- GPU: CUDA, cuDNN, (+nccl for single-node multi-gpu)
- CPU: BLAS implementation (ATLAS, MKL, or OpenBlas)
- tested configuration: BVLC/Caffe 1.0.0, CUDA 8.0, cuDNN 6.0

- Intel branch of original project
- prerequisites: mkl-dnn (based on mklml)
- tested configuration: intel/Caffe 1.0.0, mkl-dnn 0.9

Projects status:

✗ BVLC/Caffe: project closed with 1.0 (18 Apr 2017), development efforts moved to Caffe2;

✓ Intel/Caffe: latest release is 1.1.0 (13 Jan 2018).

theano

- Montreal Institute for Learning Algorithms (MILA)
- Python API (dynamic C code generation)
- prerequisites: CUDA, cuDNN, libgpuarray
- tested configuration: MILA/Theano (git: 12/07/2017), CUDA 8.0, cuDNN 6.0, libgpuarray 0.6.8

- Montreal Institute for Learning Algorithms (MILA)
- Python API (dynamic C code generation)
- prerequisites: CUDA, cuDNN, libgpuarray
- tested configuration: MILA/Theano (git: 12/07/2017), CUDA 8.0, cuDNN 6.0, libgpuarray 0.6.8

- Intel branch of original project
- prerequisites: MKL
- tested configuration: intel/Theano 1.1, MKL 2017

- Montreal Institute for Learning Algorithms (MILA)
- Python API (dynamic C code generation)
- prerequisites: CUDA, cuDNN, libgpuarray
- tested configuration: MILA/Theano (git: 12/07/2017), CUDA 8.0, cuDNN 6.0, libgpuarray 0.6.8

- Intel branch of original project
- prerequisites: MKL
- tested configuration: intel/Theano 1.1, MKL 2017

Projects status:

✗ Mila/Theano: development closed, latest bug fix is 1.0.1 (7 Dec 2017);

? Intel/Theano: latest release is 1.1.0 (1 Apr 2017).

- Intel (previously Nervana Systems)
- Python API
- prerequisites: CUDA
- tested configuration: Neon 2.1.0, CUDA 8.0, (mklml_lnx_2018)

- Intel (previously Nervana Systems)
- Python API
- prerequisites: CUDA
- tested configuration: Neon 2.1.0, CUDA 8.0, (mklml_lnx_2018)

Projects status:

✓ Nervana/Neon: latest release is 2.6.0 (5 Jan 2018).

- Google
- Python API
- prerequisites: CUDA,
- tested configuration:
  - CPU: TensorFlow (git: 18/07/2017), (mklml_lnx_2018)
  - GPU: Tensorflow 1.2.1, CUDA 8.0

https://software.intel.com/en-us/articles/tensorflow-optimizations-on-modern-intel-architecture

- Google
- Python API
- prerequisites: CUDA,
- tested configuration:
  - CPU: TensorFlow (git: 18/07/2017), (mklml_lnx_2018)
  - GPU: Tensorflow 1.2.1, CUDA 8.0

Project status:

✓ tensorflow/tensorflow: latest release 1.6.0 (28 Feb 2018)

---

https://software.intel.com/en-us/articles/tensorflow-optimizations-on-modern-intel-architecture

Software summary

CPU	GPU
Intel/Caffe	BVLC/Caffe
Intel/Theano	MILA/Theano
TensorFlow	
Neon	

Single node benchmark settings

Network structures are simplified:

- no preprocessing phase is implemented;
- input data is in-memory generated;
- no dropout is considered;
- classification task: 1000 classes.

Single node benchmark settings

Network structures are simplified:

- no preprocessing phase is implemented;
- input data is in-memory generated;
- no dropout is considered;
- classification task: 1000 classes.

Network source codes:

- Caffe/Theano/Tensorflow: convnet-benchmarks source code is used;
- Neon: code is in Neon sources.

Single node benchmark settings

Network structures are simplified:

- no preprocessing phase is implemented;
- input data is in-memory generated;
- no dropout is considered;
- classification task: 1000 classes.

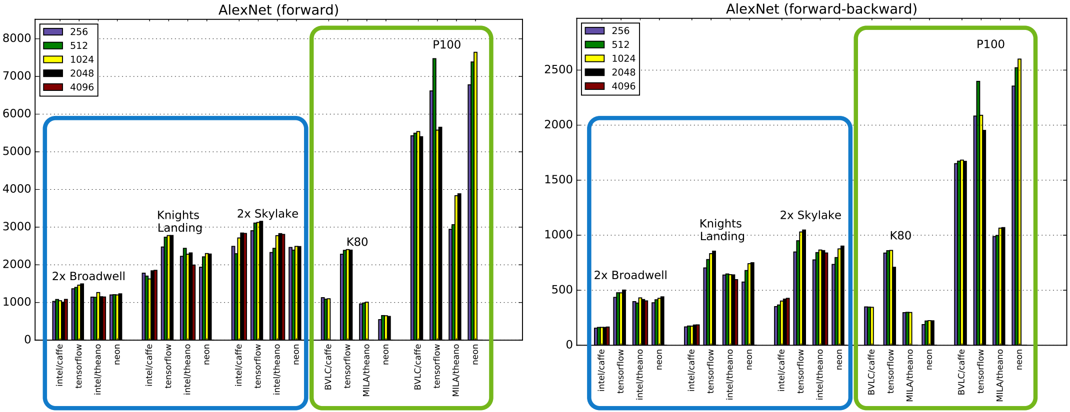
Network source codes:

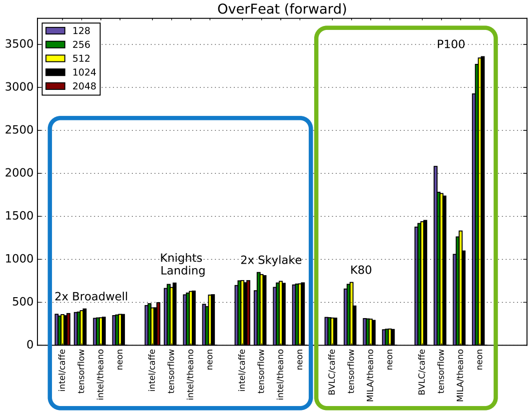
- Caffe/Theano/Tensorflow: convnet-benchmarks source code is used;
- Neon: code is in Neon sources.

Chosen performance measure: number of images per second, considering:

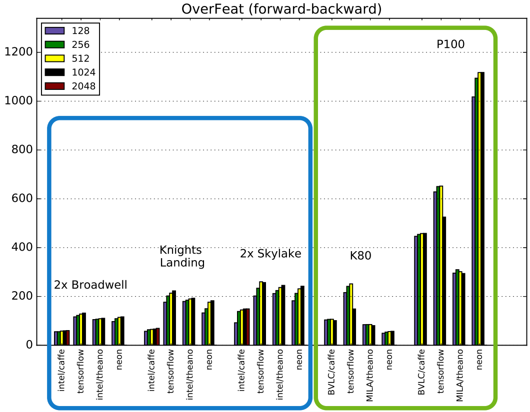
- forward step: the evaluation of the network on a batch (inference);
- forward-backward step: fwd step + the backpropagation of the errors on a batch (training).

AlexNet


AlexNet (forward)
256
512
1024
2048
4096
8000
7000
6000
5000
4000
3000
2000
1000
0
2x Broadwell
Knights Landing
2x Skylake
K80
P100
intel/caffe
tensorflow
intel/theano
neon
BVLC/caffe
MILA/theano
AlexNet (forward-backward)
2500
1500
500

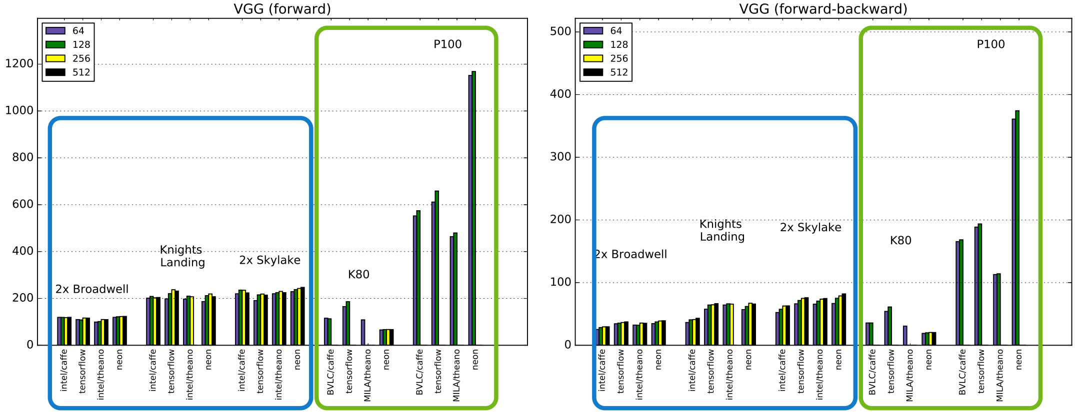


OverFeat (forward)
128
256
512
1024
2048
3500
3000
2500
2000
1500
1000
500
0
P100
K80
2x Skylake
Knights Landing
2x Broadwell
intel/caffe
tensorflow
intel/theano
neon
BVLC/caffe
MILA/theano

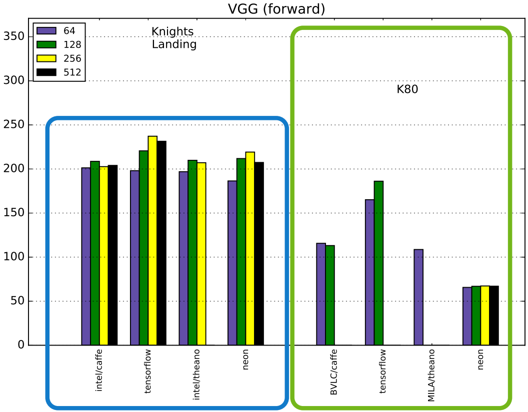


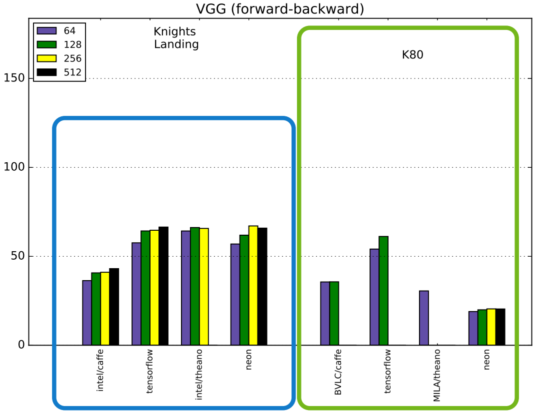
OverFeat (forward-backward)
128
256
512
1024
2048
1200
1000
800
600
400
200
0
P100
K80
2x Skylake
Knights Landing
2x Broadwell
intel/caffe
tensorflow
intel/theano
neon
BVLC/caffe
MILA/theano

VGG (I)


VGG (forward)
64
128
256
512
1200
1000
800
600
400
200
0
P100
Knights Landing
2x Skylake
K80
2x Broadwell
intel/caffe
tensorflow
intel/theano
neon
BVLC/caffe
MILA/theano
VGG (forward-backward)
500
400
300
200
100
0

VGG (II)


VGG (forward)
64
128
256
512
Knights Landing
K80
350
300
250
200
150
100
50
0
intel/caffe
tensorflow
intel/theano
neon
BVLC/caffe
tensorflow
MILA/theano
neon


VGG (forward-backward)
64
128
256
512
Knights Landing
K80
150
100
50
0
intel/caffe
tensorflow
intel/theano
neon
BVLC/caffe
tensorflow
MILA/theano
neon

Based on official TensorFlow benchmark suite: https://github.com/tensorflow/benchmarks

- input data can be in-memory generated, or from dataset (+preprocessing);
- contains definitions of fully functional neural networks (dropout is present);
- support for multinode (and multinode, multi-GPU runs);
- support for larger (and growing) number of models;

but the support is for TensorFlow only.

Goyal et al., ***Accurate, Large Minibatch SGD Training ImageNet in 1 Hour*****, 2017**

Based on official TensorFlow benchmark suite: https://github.com/tensorflow/benchmarks

- input data can be in-memory generated, or from dataset (+preprocessing);
- contains definitions of fully functional neural networks (dropout is present);
- support for multinode (and multinode, multi-GPU runs);
- support for larger (and growing) number of models;

but the support is for TensorFlow only.

Multi node parallelization approach: given $n = \#\text{batch}, M = \#\text{nodes}$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \epsilon_k \hat{\boldsymbol{g}}^{(k)} \qquad \text{where} \qquad \hat{\boldsymbol{g}}^{(k)} = \frac{1}{Mn} \nabla_{\boldsymbol{\theta}} \sum_{m \in \text{nodes}} \sum_{l \in \text{batch}_m} L(f(\boldsymbol{x}^{(l)}; \boldsymbol{\theta}^{(k)}), \boldsymbol{y}^{(l)})$$

Goyal et al., ***Accurate, Large Minibatch SGD Training ImageNet in 1 Hour***, **2017**

Supported communication protocols

TensorFlow supported protocols:

- gRPC: google Remote Procedure Call
- gRPC+VERBS:
  - gRPC for administrative tasks (set up RDMA path),
  - RDMA (Remote Direct Memory Access) for actual tensors (weights, gradients, etc) exchange.
- gRPC+MPI: [not tested yet]

---

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/verbs/README.md

Supported communication protocols

TensorFlow supported protocols:

- gRPC: google Remote Procedure Call
- gRPC+VERBS:
  - gRPC for administrative tasks (set up RDMA path),
  - RDMA (Remote Direct Memory Access) for actual tensors (weights, gradients, etc) exchange.
- gRPC+MPI: [not tested yet]

Supported variable management procedures:

- parameter server: variables are stored on a parameter server that holds the master copy of the variable. For each step, each wn gets a copy of the variables from the ps, and sends its gradients to the ps;
- distributed replicated: wn has a copy of the variables, and updates its copy after the ps are all updated with the gradients from all wn [not tested yet].

---

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/verbs/README.md

Supported communication protocols

TensorFlow supported protocols:

- gRPC: google Remote Procedure Call
- gRPC+VERBS:
  - gRPC for administrative tasks (set up RDMA path),
  - RDMA (Remote Direct Memory Access) for actual tensors (weights, gradients, etc) exchange.
- gRPC+MPI: [not tested yet]

Supported variable management procedures:

- parameter server: variables are stored on a parameter server that holds the master copy of the variable. For each step, each wn gets a copy of the variables from the ps, and sends its gradients to the ps;
- distributed replicated: wn has a copy of the variables, and updates its copy after the ps are all updated with the gradients from all wn [not tested yet].

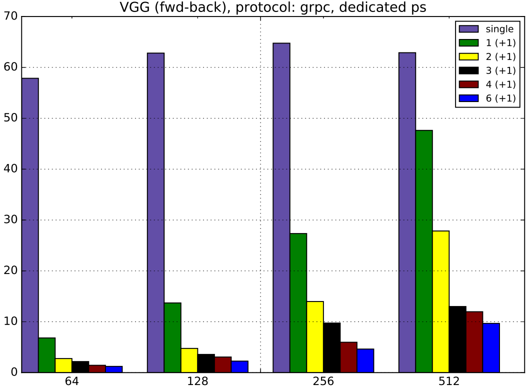
Chosen performance measure: average number of images per second per node, considering:

- fwd step + the backpropagation of the errors on a batch (training);
- fixed local batch size comparison: overall batch size grows with nodes.

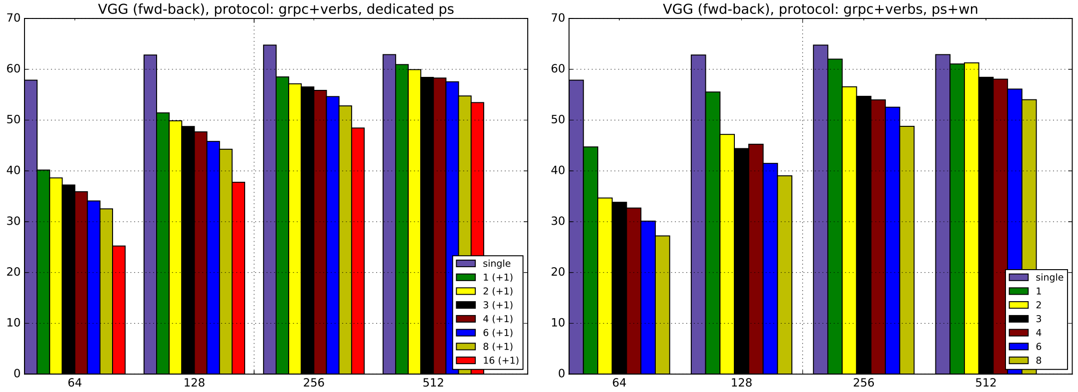
---

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/verbs/README.md

gRPC (forward-backward,VGG)



gRPC+VERBS (forward-backward,VGG)



Single node benchmarks:

- Best combination is Nvidia P100 and Neon.
- Nodes composed by a single Intel Knights Landing or 2x intel Skylake can achieve comparable, or slightly superior performances to a single Nvidia K80.
- Concerning CPU systems, no software outperforms the others.
- TensorFlow exhibit good performances in both architectures.

Single node benchmarks:

- Best combination is Nvidia P100 and Neon.
- Nodes composed by a single Intel Knights Landing or 2x intel Skylake can achieve comparable, or slightly superior performances to a single Nvidia K80.
- Concerning CPU systems, no software outperforms the others.
- TensorFlow exhibit good performances in both architectures.

Multinode benchmarks (TensorFlow only):

- RDMA (gRPC+VERBS) is fundamental for acceptable efficiency.

Single node benchmarks:

- Best combination is Nvidia P100 and Neon.
- Nodes composed by a single Intel Knights Landing or 2x intel Skylake can achieve comparable, or slightly superior performances to a single Nvidia K80.
- Concerning CPU systems, no software outperforms the others.
- TensorFlow exhibit good performances in both architectures.

Multinode benchmarks (TensorFlow only):

- RDMA (gRPC+VERBS) is fundamental for acceptable efficiency.

Ongoing work:

- Caffe and Theano can exploit Intel Machine Learning Scaling Library (MLSL);
- TensorFlow can exploit also MPI.

References and Acknowledgements

References:

- Russakovsky et al., *ImageNet Large Scale Visual Recognition Challenge*, 2015
- Krizhevsky, Sutskever, Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, 2012
- Sermanet et al., *Overfeat: Integrated recognition, localization and detection using convolutional networks*, 2014
- Simonyan, Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014
- Goyal et al., *Accurate, Large Minibatch SGD Training ImageNet in 1 Hour*, 2017
- https://software.intel.com/en-us/articles/tensorflow-optimizations-on-modern-intel-architecture
- https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/verbs/README.md

We would like to thank:

- *Andrea Luiselli (Intel)*, for support on new Intel software branches;
- *Walter Riviera (Intel)*, for multinode TensorFlow support.

AlexNet	OverFeat	VGG_a (vgg11)
input RGB image		
conv: k 11×11, ch 64, stride 4 maxpool: k 3×3, stride 2	conv: k 11×11, ch 96, stride 4 maxpool: k 2×2, stride 2	conv: k 3×3, ch 64, stride 1 maxpool: k 2×2, stride 2
conv: k 5×5, ch 192, stride 1 maxpool: k 3×3, stride 2	conv: k 5×5, ch 256, stride 1 maxpool: k 2×2, stride 2	conv: k 3×3, ch 128, stride 1 maxpool: k 2×2, stride 2
conv: k 3×3, ch 384, stride 1	conv: k 3×3, ch 512, stride 1	conv: k 3×3, ch 256, stride 1 conv: k 3×3, ch 256, stride 1 maxpool: k 2×2, stride 2
conv: k 3×3, ch 256, stride 1	conv: k 3×3, ch 1024, stride 1	conv: k 3×3, ch 512, stride 1 conv: k 3×3, ch 512, stride 1 maxpool: k 2×2, stride 2
conv: k 3×3, ch 256, stride 1 maxpool: k 3×3, stride 2	conv: k 3×3, ch 1024, stride 1 maxpool: k 2×2, stride 2	conv: k 3×3, ch 512, stride 1 conv: k 3×3, ch 512, stride 1 maxpool: k 2×2, stride 2
FC: output 4096	FC: output 3072	FC: output 4096
FC: output 4096	FC: output 4096	FC: output 4096
FC: output 1000	FC: output 1000	FC: output 1000

Table: Convolutional and fully-connected layers are followed by ReLU nonlinear function.

Reviewers feedbacks

- Not familiar with applications, so the legend on the graphs is unclear. Unless you want to compare across FWD and FWD-BWD, I would put the graphs on separate slides so that you can increase their sizes. I hope the talk includes a brief discussion about how the applications are different. It isn't clear how the work is being distributed and there are many gaps in the results.

- This seems a nice benchmark for a relevant case study. I suggest to improve the quality of the slide 6: what is on the y axis? Too many columns make the comparison difficult: I suggest to split the plots. On slide 7: capitalize Intel;

Reviewers questions

- **You state that only TensorFlow supports MPI. Does that mean that all of the applications are running OpenMP-only for this work?**
  Single-node benckmark results are obtained with applications running OpenMP-only. Multinode TensorFlow resides on gRPC or gRPC+VERBS.
- **Why aren't there results for 4096 NEON? Why are the GPU results missing 4096? Why are half of the GPU results missing 2048?**
  K80 has a total of 24 GB of memory: for large batch sizes, all SDKs runs out of memory.
- **Were any of the results surprising?**
  Intel efforts on CPU based Deep Learning allows TensorFlow to request comparable cpu time either on KNL or on K80.
- **Are there remaining challenges or questions to be answered?**
- **Comment on the parallelization of Neon: multithreading?**
  All tested SDKs are exploiting multitreading parallelization.