DEEP LEARNING FRAMEWORKS REVIEW

한재근 과장 | jahan@nvidia.com
유현곤 부장 | hryu@nvidia.com / 양한별 과장 | hanbyuly@nvidia.com

NVIDIA

AGENDA

Sequentials

Why Tensorflow is slower than others

Sequentials

Caffe script for Model

Simple CNN

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "scale"
  top: "conv1"
  param {
    lr_mult: 0.0
  }
  param {
    lr_mult: 0.0
  }
convolution_param {
    num_output: 20
    kernel_size: 5
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
```

```
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 2
    stride: 2
  }
}
```

KERAS sequentials

Simple ANN

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation

model = Sequential()

model.add(Dense(64, input_dim=20, init='uniform'))
model.add(Activation('tanh'))
model.add(Dropout(0.5))
model.add(Dense(64, init='uniform'))
model.add(Activation('tanh'))
model.add(Dropout(0.5))
model.add(Dense(10, init='uniform'))
model.add(Activation('softmax'))
```

KERAS sequentials

Simple CNN

```
model = Sequential()
model.add(Convolution2D(32, 3, 3, border_mode='valid', input_shape=(3, 100, 100))) model.add(Activation('relu'))
model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 3, 3, border_mode='valid'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(10))
model.add(Activation('softmax'))
```

MXNET sequentials : symbols

alexnet

```
def get_symbol(num_classes, **kwargs):
  input_data = mx.symbol.Variable(name="data")

  conv1 = mx.symbol.Convolution( data=input_data, kernel=(11, 11), stride=(4, 4), num_filter=96)
  relu1 = mx.symbol.Activation(data=conv1, act_type="relu")
  pool1 = mx.symbol.Pooling( data=relu1, pool_type="max", kernel=(3, 3), stride=(2,2))
  lrn1 = mx.symbol.LRN(data=pool1, alpha=0.0001, beta=0.75, knorm=1, nsize=5)

  conv2 = mx.symbol.Convolution( data=lrn1, kernel=(5, 5), pad=(2, 2), num_filter=256)
  relu2 = mx.symbol.Activation(data=conv2, act_type="relu")
  pool2 = mx.symbol.Pooling(data=relu2, kernel=(3, 3), stride=(2, 2), pool_type="max")
  lrn2 = mx.symbol.LRN(data=pool2, alpha=0.0001, beta=0.75, knorm=1, nsize=5)

  conv3 = mx.symbol.Convolution( data=lrn2, kernel=(3, 3), pad=(1, 1), num_filter=384)
  relu3 = mx.symbol.Activation(data=conv3, act_type="relu")
  conv4 = mx.symbol.Convolution( data=relu3, kernel=(3, 3), pad=(1, 1), num_filter=384)
  relu4 = mx.symbol.Activation(data=conv4, act_type="relu")
  conv5 = mx.symbol.Convolution( data=relu4, kernel=(3, 3), pad=(1, 1), num_filter=256)
  relu5 = mx.symbol.Activation(data=conv5, act_type="relu")
  pool3 = mx.symbol.Pooling(data=relu5, kernel=(3, 3), stride=(2, 2), pool_type="max")
  flatten = mx.symbol.Flatten(data=pool3)

  fc1 = mx.symbol.FullyConnected(data=flatten, num_hidden=4096)
  relu6 = mx.symbol.Activation(data=fc1, act_type="relu")
  dropout1 = mx.symbol.Dropout(data=relu6, p=0.5)
```

Torch Sequentials

Simple ANN

```
nn.Sequential()

local lenet = nn.Sequential()
lenet:add(nn.MulConstant(0.0125))
lenet:add(backend.SpatialConvolution(channels,20,5,5,1,1,0))
lenet:add(backend.SpatialMaxPooling(2, 2, 2, 2))
lenet:add(backend.SpatialConvolution(20,50,5,5,1,1,0))
lenet:add(backend.SpatialMaxPooling(2,2,2,2))
lenet:add(nn.View(-1):setNumInputDims(3))
lenet:add(nn.Linear(800,500))
lenet:add(backend.ReLU())
lenet:add(nn.Linear(500, nclasses))
lenet:add(nn.LogSoftMax())
```

Troch Inception

```
features:add(nn.MulConstant(0.02))
  features:add(convLayer(nChannels,64,7,7,2,2,3,3)):add(backend.SpatialBatchNormalization(64,1e-3)):add(backend.ReLU(true))
  features:add(backend.SpatialMaxPooling(3,3,2,2):ceil())
  features:add(convLayer(64,64,1,1)):add(backend.SpatialBatchNormalization(64,1e-3)):add(backend.ReLU(true))
  features:add(convLayer(64,192,3,3,1,1,1,1)):add(backend.SpatialBatchNormalization(192,1e-3)):add(backend.ReLU(true))
  features:add(backend.SpatialMaxPooling(3,3,2,2):ceil())
  features:add(inception( 192, {{ 64},{ 64, 64},{ 64, 96},{'avg', 32}})) -- 3(a)
  features:add(inception( 256, {{ 64},{ 64, 96},{ 64, 96},{'avg', 64}})) -- 3(b)
  features:add(inception( 320, {{  0},{128,160},{ 64, 96},{'max',  0}})) -- 3(c)
  features:add(convLayer(576,576,2,2,2,2)):add(backend.SpatialBatchNormalization(576,1e-3))
  features:add(inception( 576, {{224},{ 64, 96},{ 96,128},{'avg',128}})) -- 4(a)
  features:add(inception( 576, {{192},{ 96,128},{ 96,128},{'avg',128}})) -- 4(b)
  features:add(inception( 576, {{160},{128,160},{128,160},{'avg', 96}})) -- 4(c)
  features:add(inception( 576, {{ 96},{128,192},{160,192},{'avg', 96}})) -- 4(d)
```

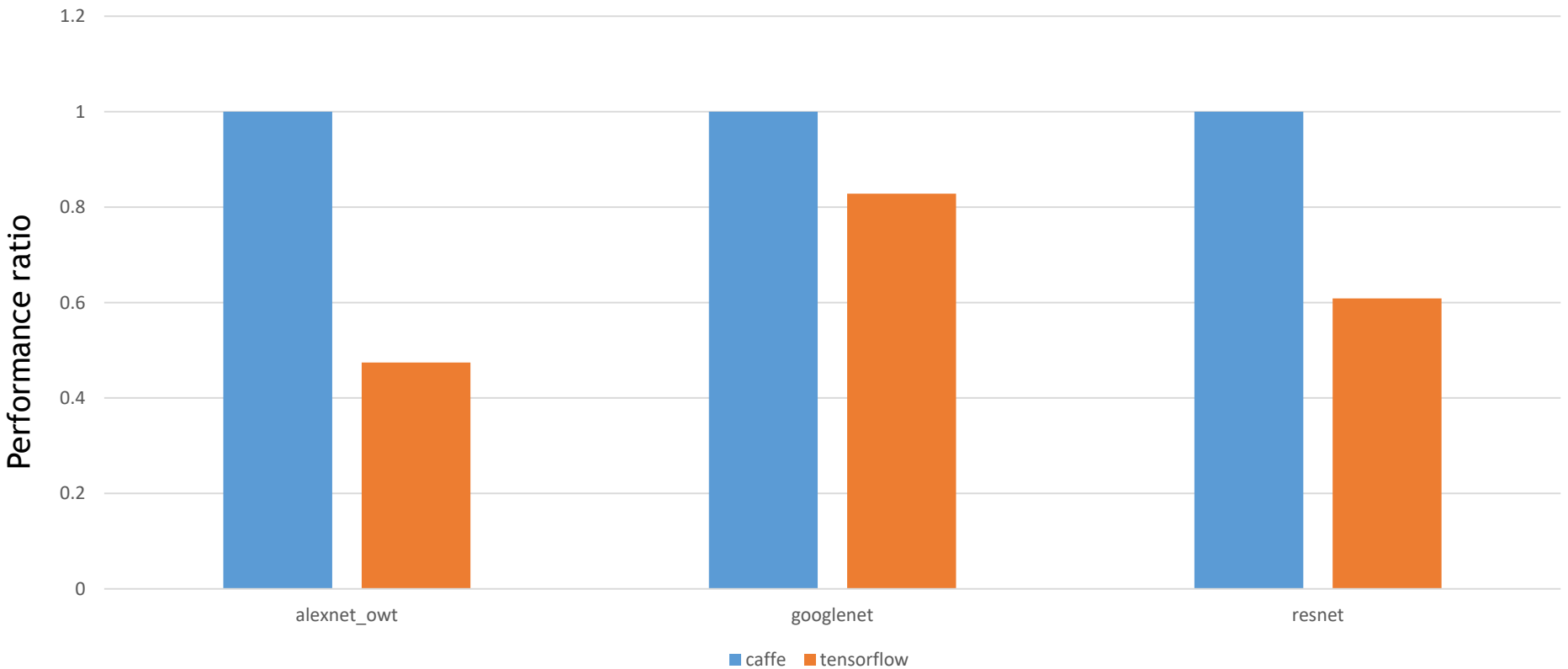
```
local function inception(input_size, config)
   local concat = nn.Concat(2)
         ...
   local conv3 = nn.Sequential()
         ...
   local conv3xx = nn.Sequential()
   local pool = nn.Sequential()
end
```

```
local conv3 = nn.Sequential()
      conv3:add(convLayer( input_size, config[2][1],1,1,1,1)):add(backend.ReLU(
      conv3:add(convLayer(config[2][1], config[2][2],3,3,1,1,1,1)):add(backend.
      concat:add(conv3
```

Why Tensorflow is slow?

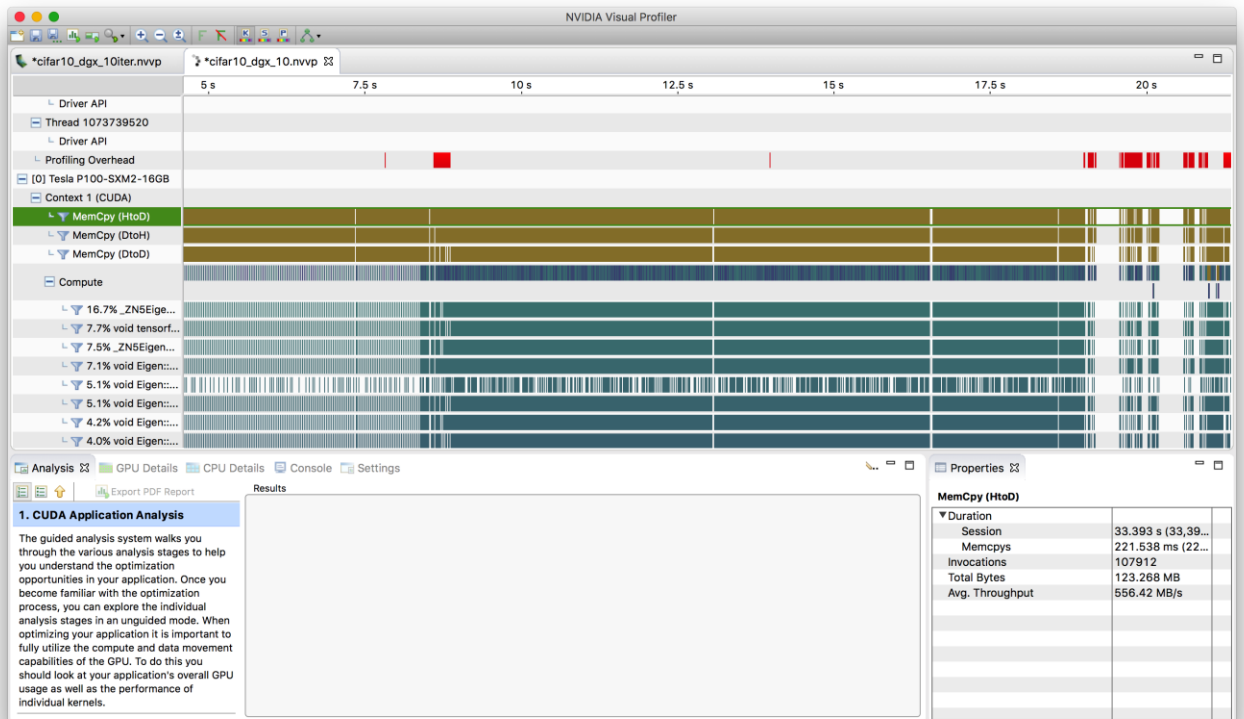
One Day…

We found that tensorflow is really slow



NVIDIA Profile Result - Tensorflow

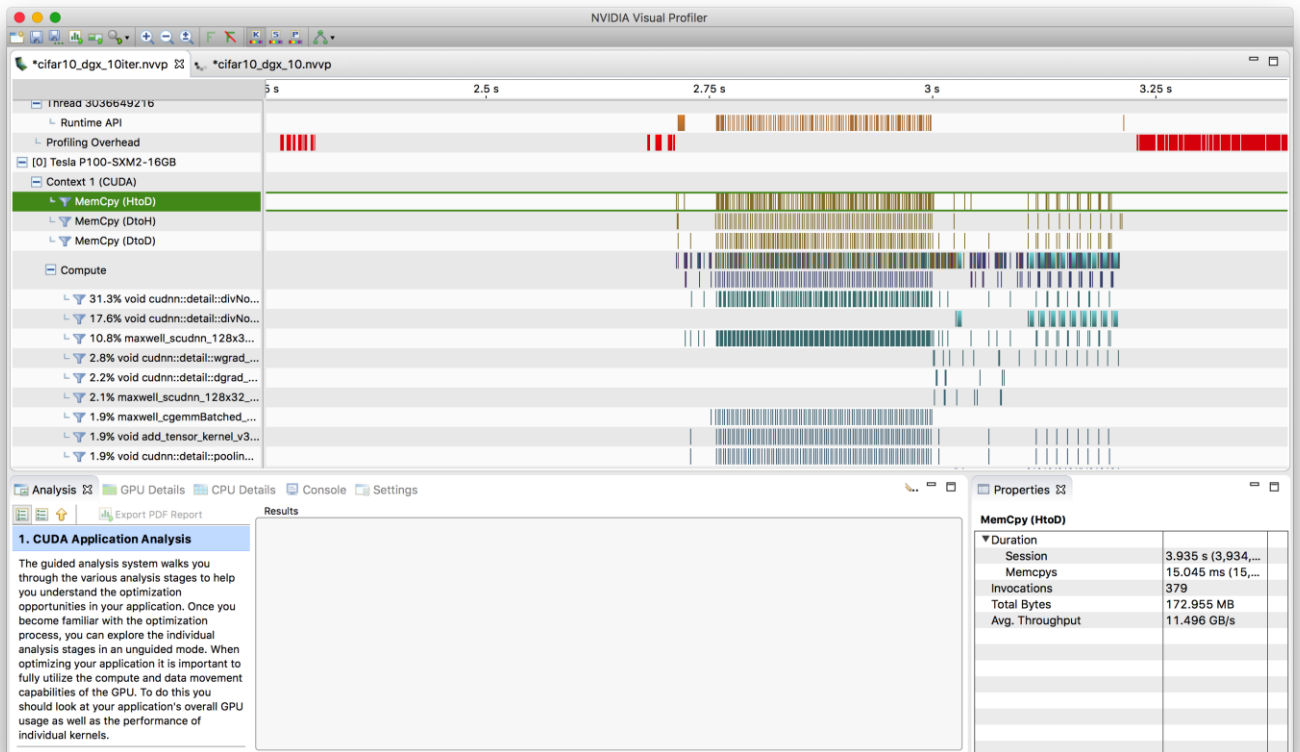
CIFAR10 example, 10 iteration



	Invocations	Total Bytes (MB)	Avg. Throughput	Duration (ms)
CPU → GPU	107912	123.256	556.42 MB/s	221.538
GPU → CPU	107883	220.186	1.455 GB	151.341
GPU → GPU	64615	258.46	3.005 MB/s	86.024

NVIDIA Profile Result - Caffe

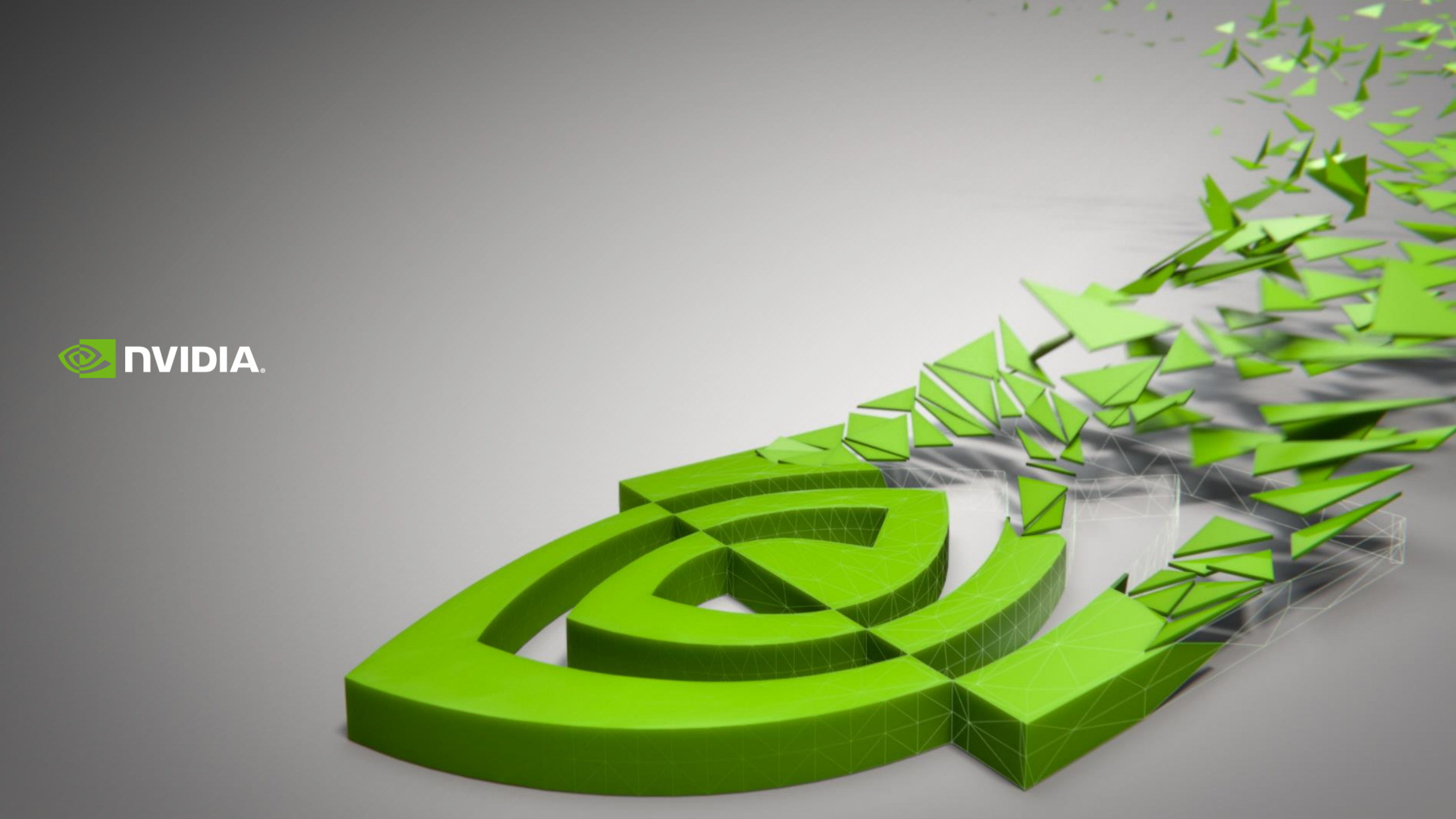
CIFAR10 example, 10 iteration



	Invocations	Total Bytes (MB)	Avg. Throughput (GB/s)	Duration (ms)
HtoD	379	172.955	11.496	15.045
DtoH	571	17.973	7.872	2.283
DtoD	450	508.779	202.713	2.51

What we found is,

- Tensorflow is GPU operation is somewhat coarse
  - GPU Utilization is low
  - Heavy CPU-GPU latency bottleneck
  - Low CPU-GPU communication bandwidth
- It mainly uses eigen3 open source library for linear algebra, not cuDNN


NVIDIA