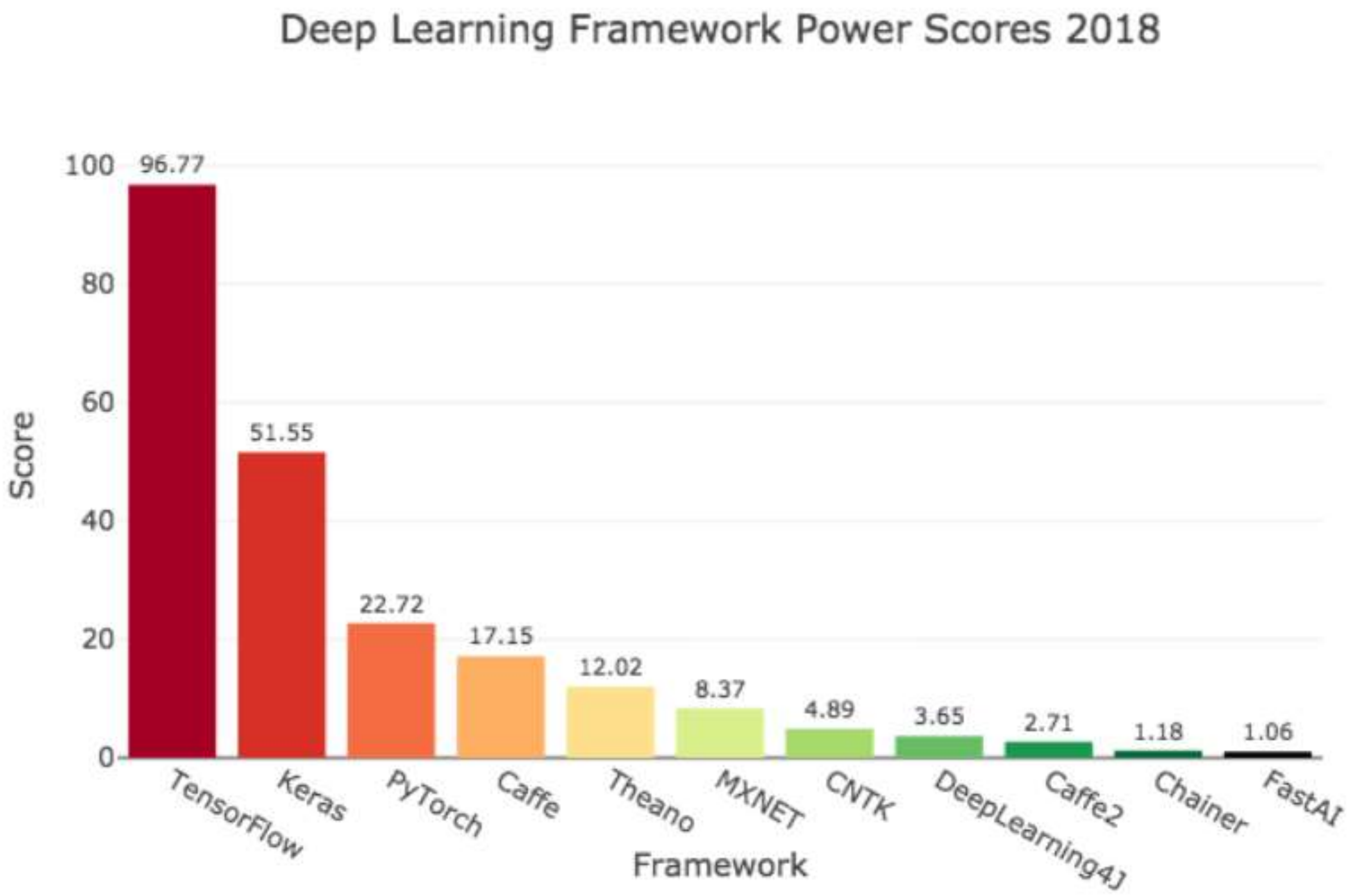
Deep Learning Frameworks

DELL EMC

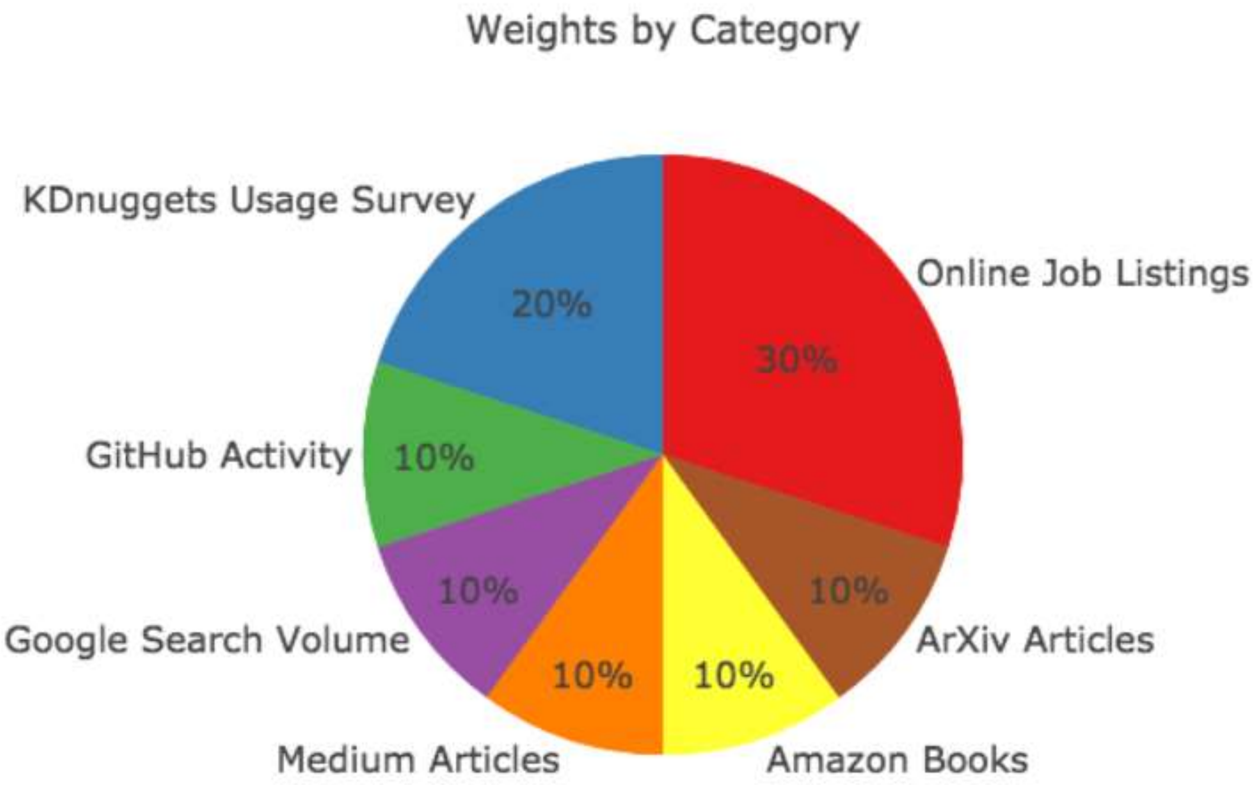
Deep Learning Frameworks

- Tensorflow: Google
- Keras: Google
- Pytorch: Facebook
- Caffe: Berkley
- Theano (discontinued)
- MXNET: Apache (Amazon)
- CNTK: Microsoft
- Caffee2: Facebook

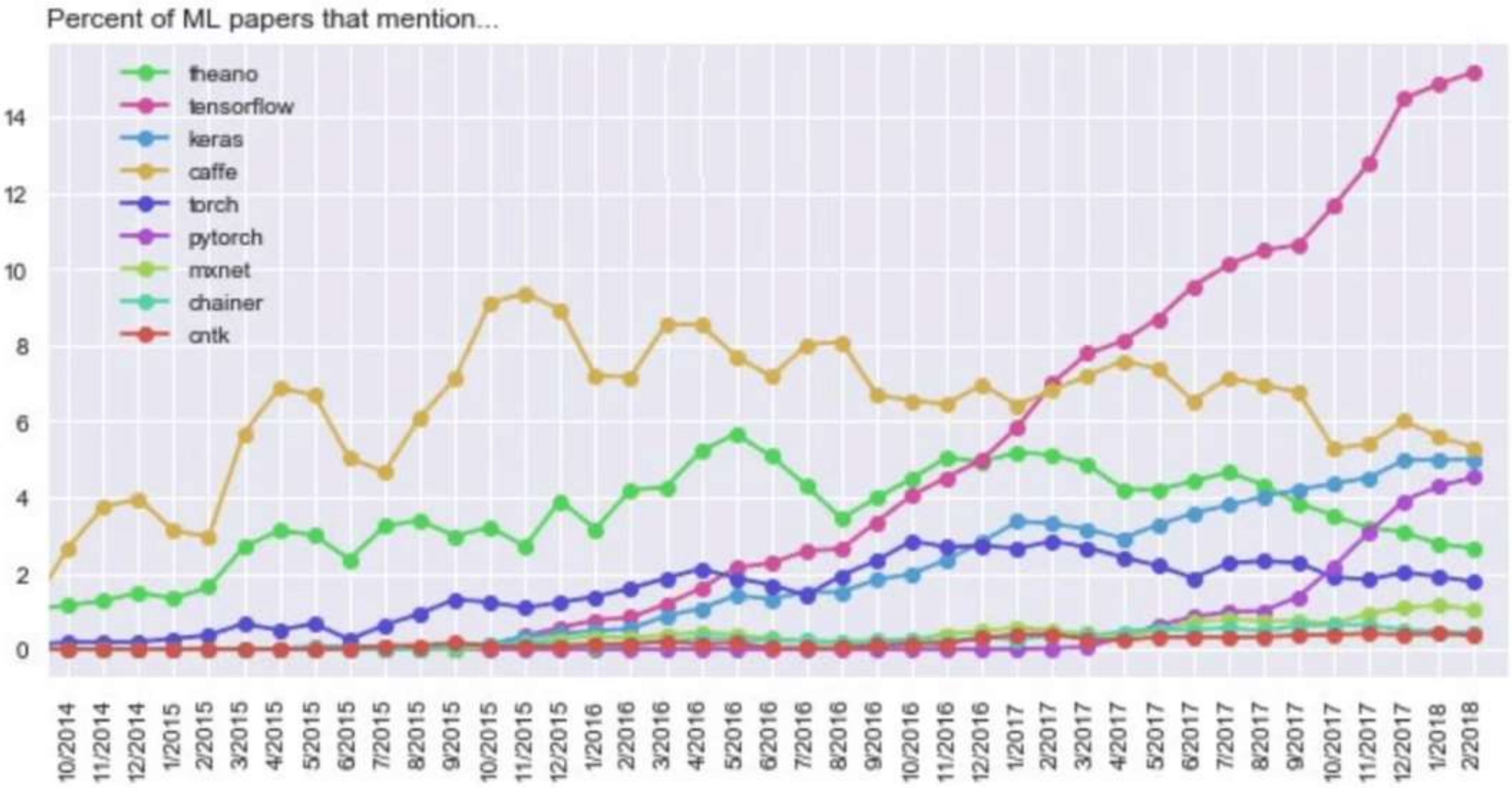


[Deep Learning Framework Power Scores 2018]

Power Score Weights



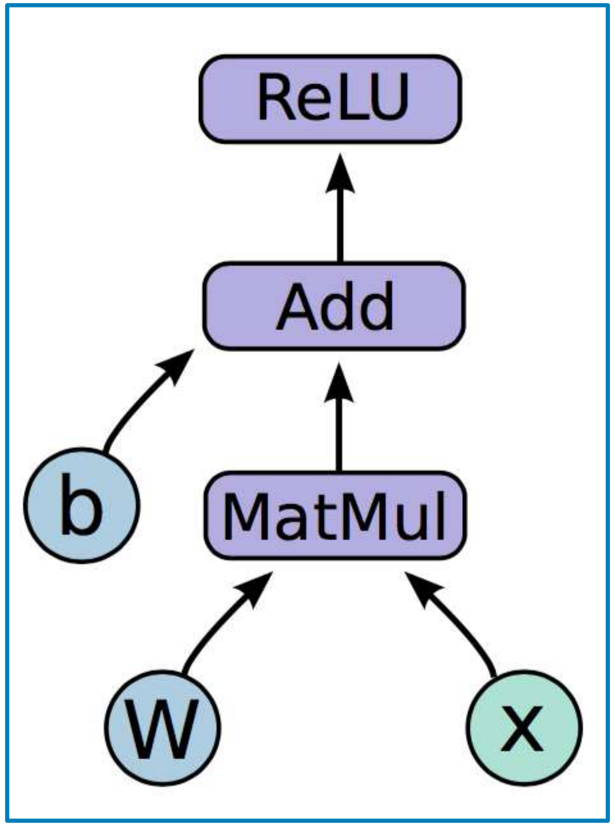
Trends Deep Learning Frameworks



[Tensorflow 2.0 Changes]

Programing Model Tensorflow

$$h_i = \mathrm{ReLU}(Wx + b)$$



- **Variables** are 0-ary stateful nodes which output their current value.
- **Placeholders** are 0-ary nodes whose value is fed in at execution time.
- **Mathematical operations:**
  - **MatMul:** Multiply two matrix values
  - **Add:** Add elementwise (with broadcasting).
  - **ReLU:** Activate with elementwise rectified linear function.

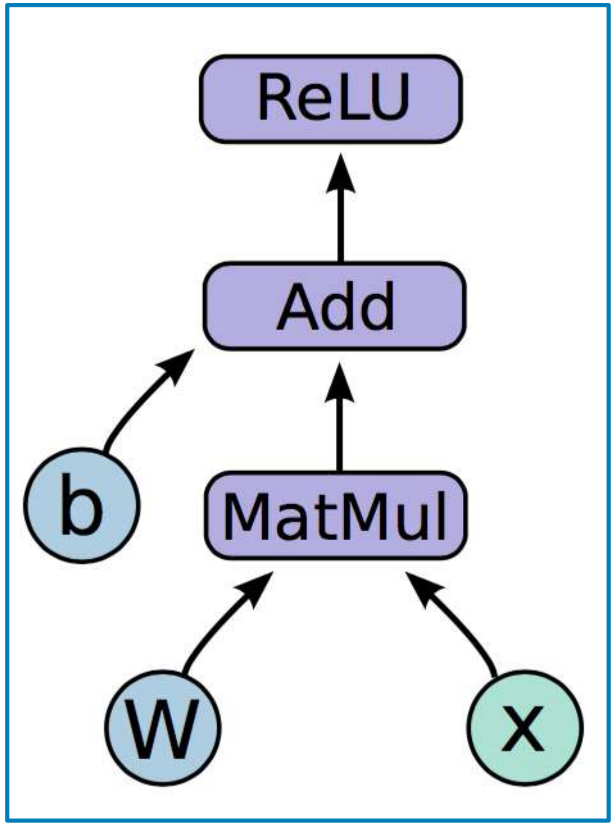
Programing Model Tensorflow

```
import numpy as np
  import tensorflow as tf

  b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784, 100),
                -1, 1))

x = tf.placeholder(tf.float32, (None, 784))
h_i = tf.nn.relu(tf.matmul(x, W) + b)


sess = tf.Session()
sess.run(tf.initialize_all_variables())
sess.run(h_i, {x: np.random.random(64, 784)})
```



Keras: Making neural Nets and Tensorflow easy

VGG-like convnet:

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D
from keras.optimizers import SGD

model = Sequential()
# input: 100x100 images with 3 channels -> (3, 100, 100) tensors.
# this applies 32 convolution filters of size 3x3 each.
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())
# Note: Keras does automatic shape inference.
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))

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

sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd)

model.fit(X_train, Y_train, batch_size=32, nb_epoch=1)
```

Eager Execution

Standard Graph Mode

```
a = tf.constant([[1, 2],
                 [3, 4]])
print(a)
```

```
Tensor("Const_1:0", shape=(2, 2), dtype=int32)
```

```
[3] with tf.Session() as sess:
      output = sess.run(a)
      print(output)
```

```
[[1 2]
 [3 4]]
```

Eager Execution

```
[27] tf.enable_eager_execution()
     a = tf.constant([[1, 2],
                      [3, 4]])
     print(a)
```

```
tf.Tensor(
[[1 2]
 [3 4]], shape=(2, 2), dtype=int32)
```

Pytorch

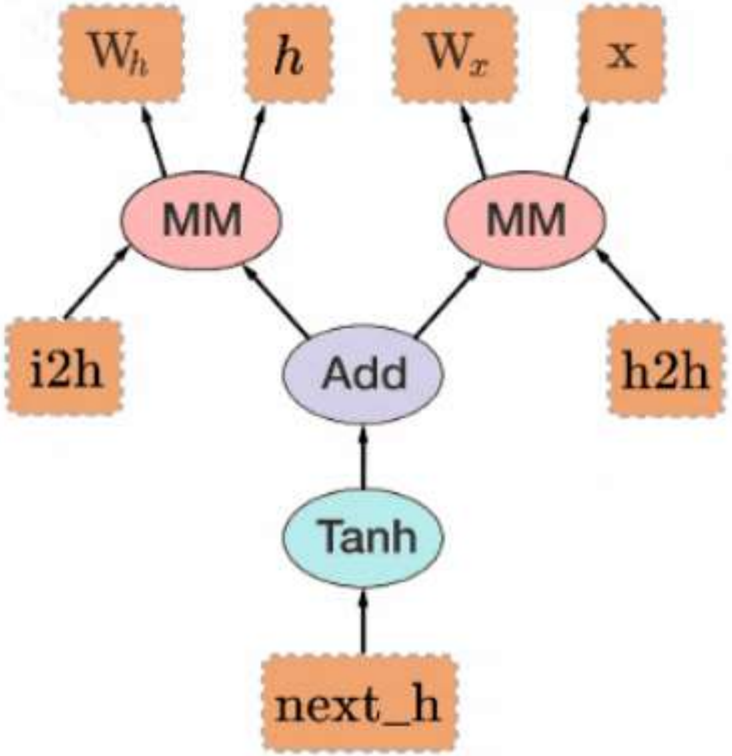
Back-propagation uses the dynamically built graph

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
next_h = i2h + h2h
next_h = next_h.tanh()

next_h.backward(torch.ones(1, 20))
```

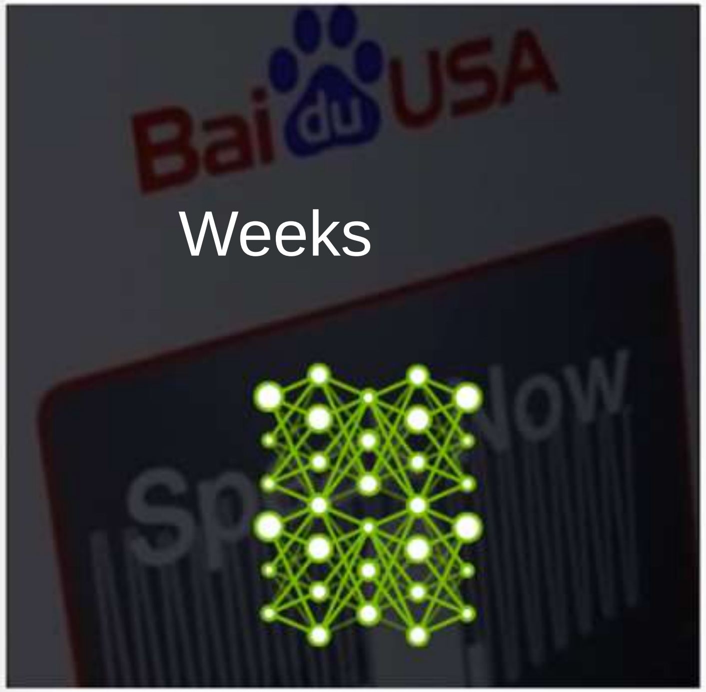


Training is the main Challenge of AI

7 ExaFLOPS
60 Million Parameters

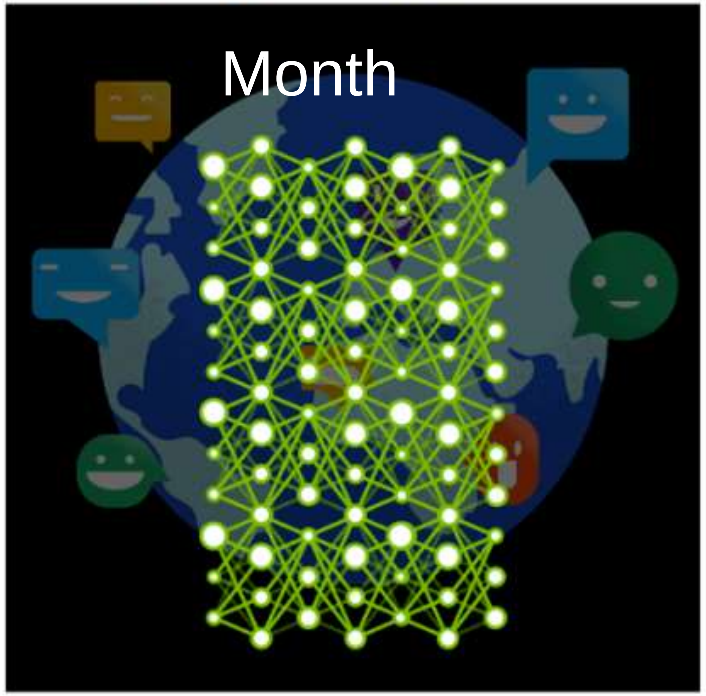
2015 - Microsoft ResNet
Superhuman Image Recognition

20 ExaFLOPS
300 Million Parameters



2016 - Baidu Deep Speech 2
Superhuman Voice Recognition

100 ExaFLOPS
8700 Million Parameters

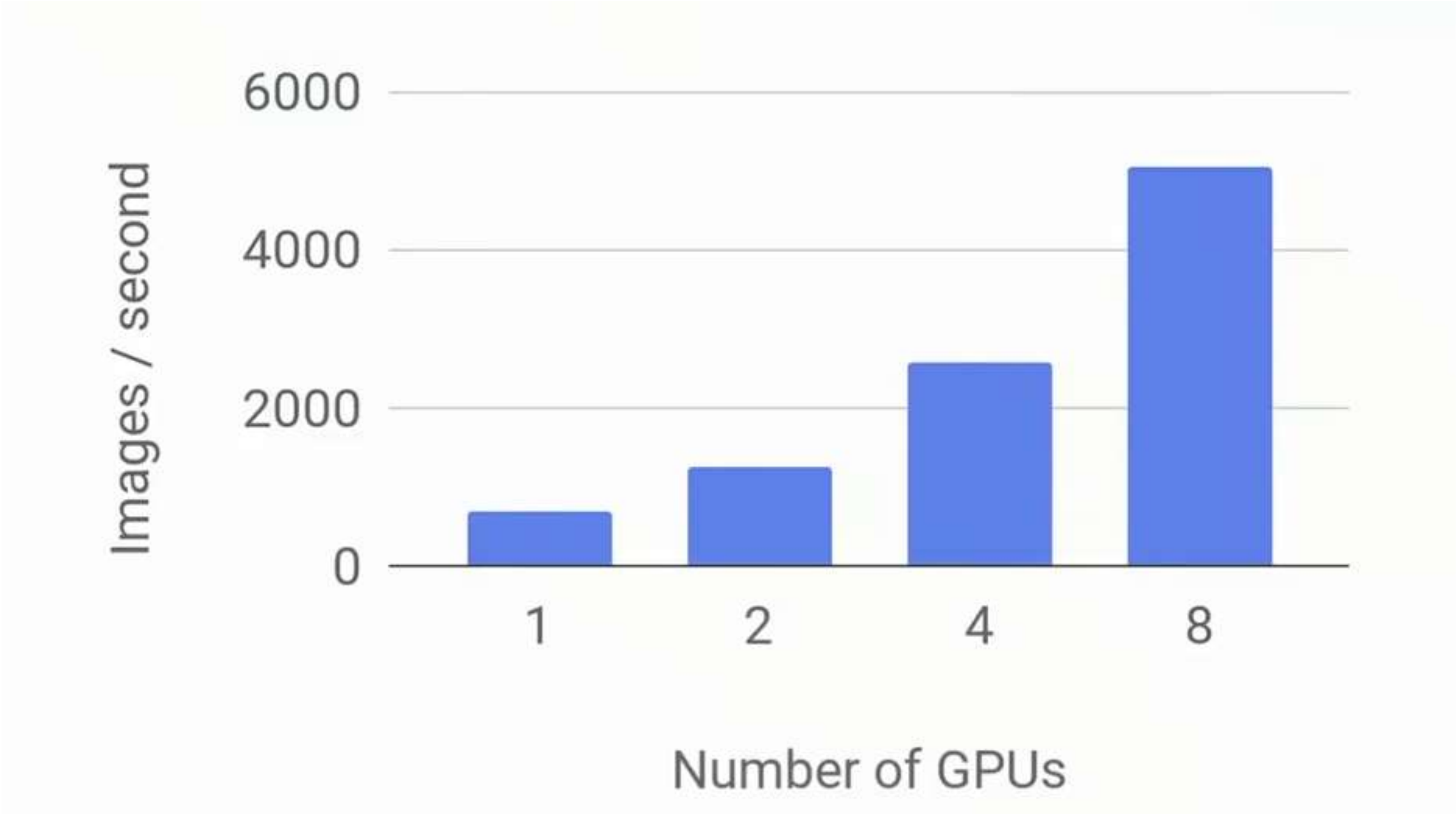


2017 - Google Neural Machine Translation
Near Human Language Translation

Courtesy NVidia

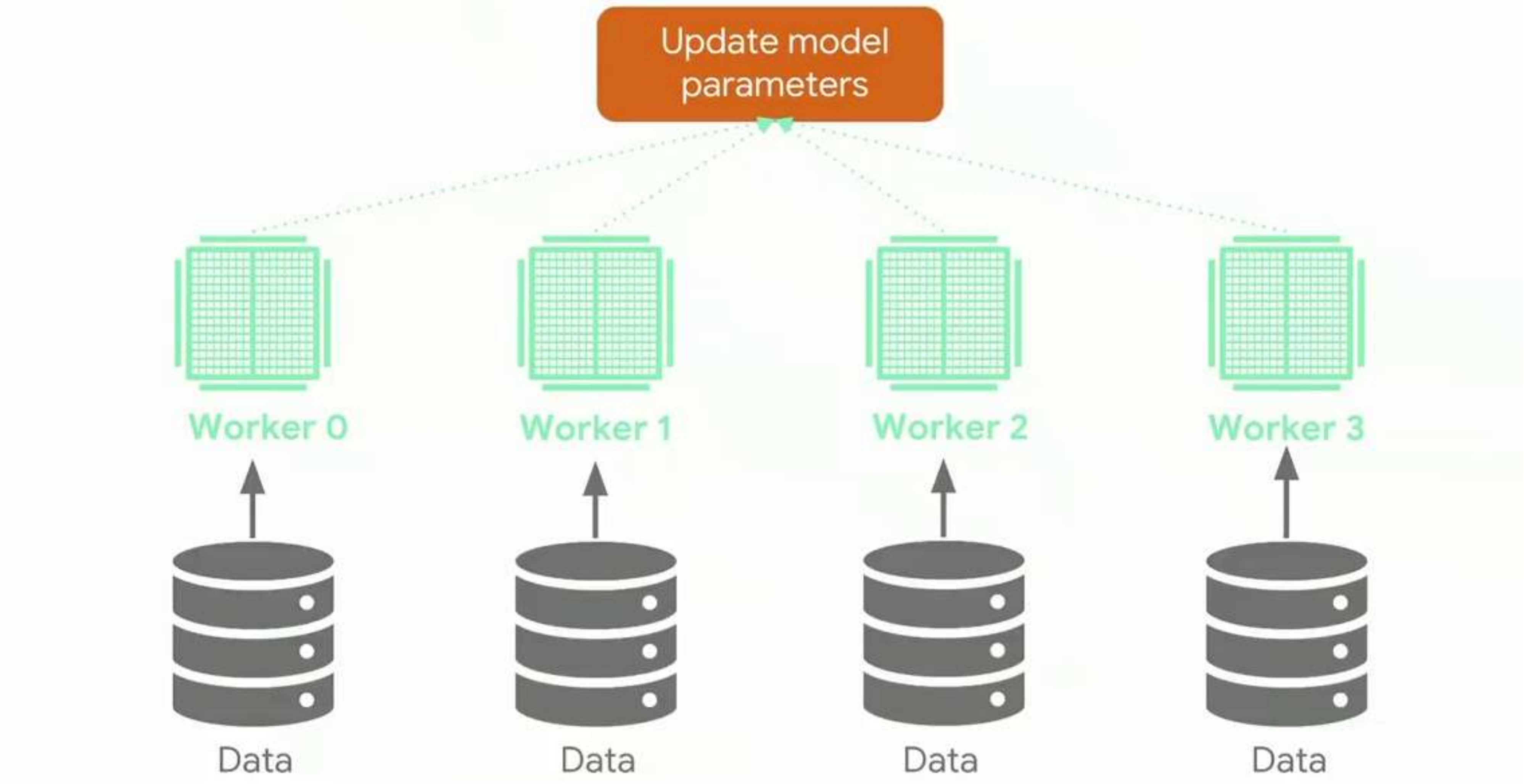
Fastest Accelerator Cards Today ~ 100 Tera Flops/sec

Scaling with distributed Training



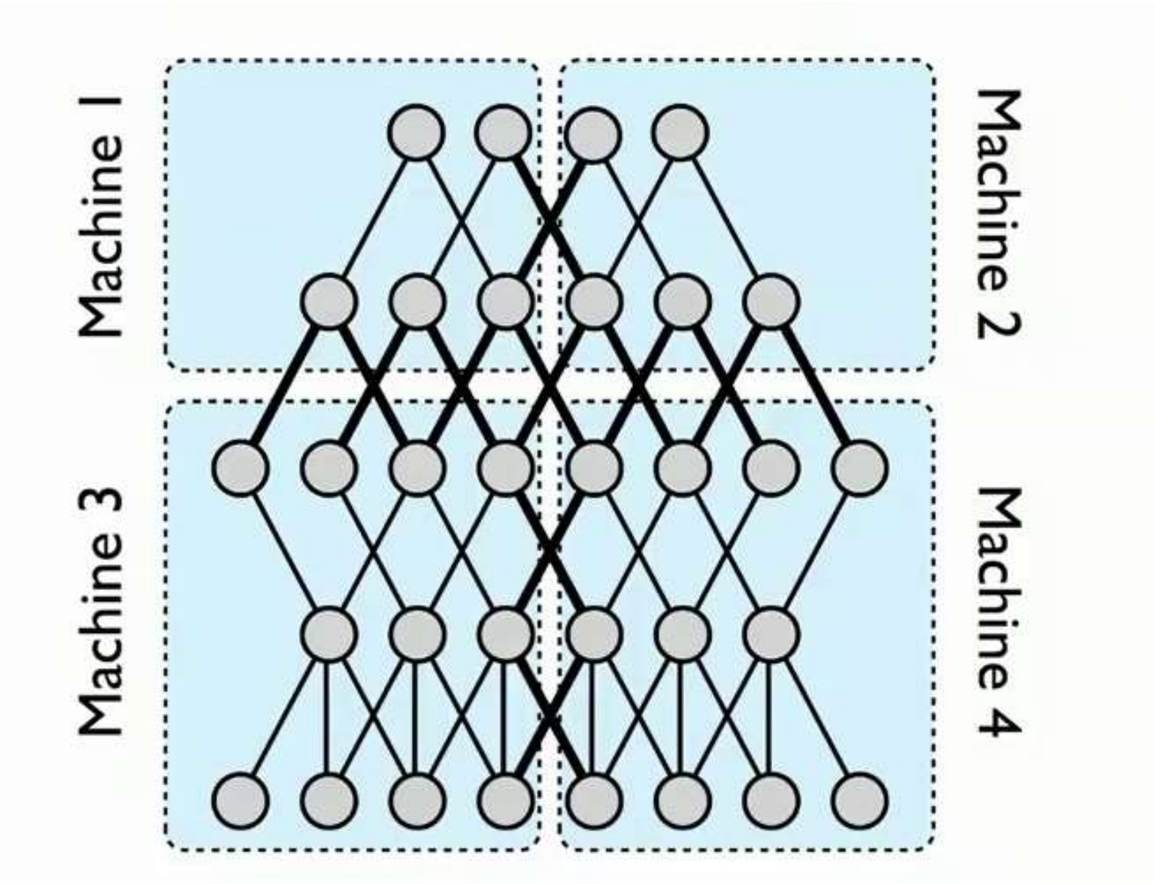
Source Google

Distributed Training: Data Parallelism



Source Google

Distributed Training: Model Parallelism

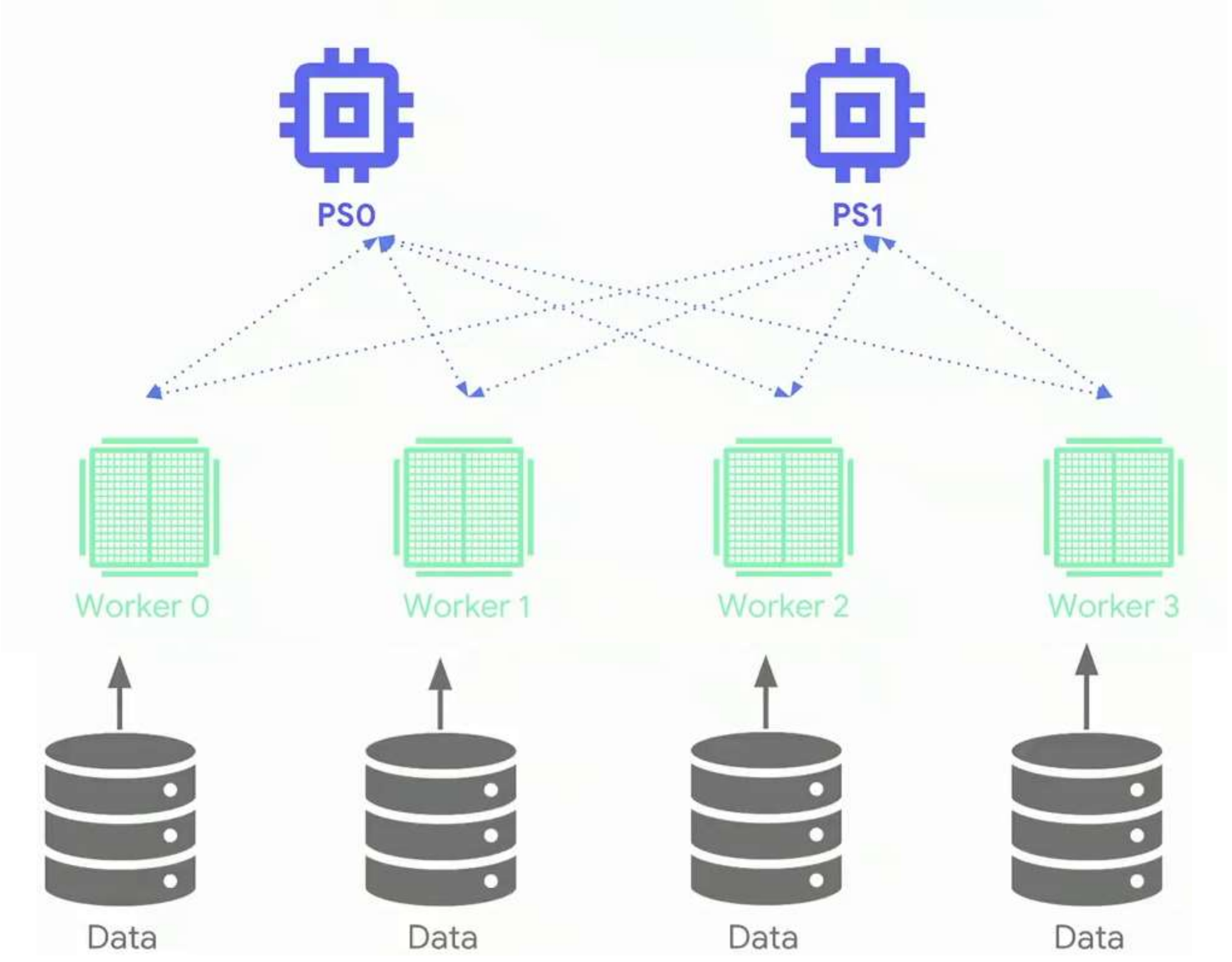


Model and Data Parallelism possible together

Rarely used

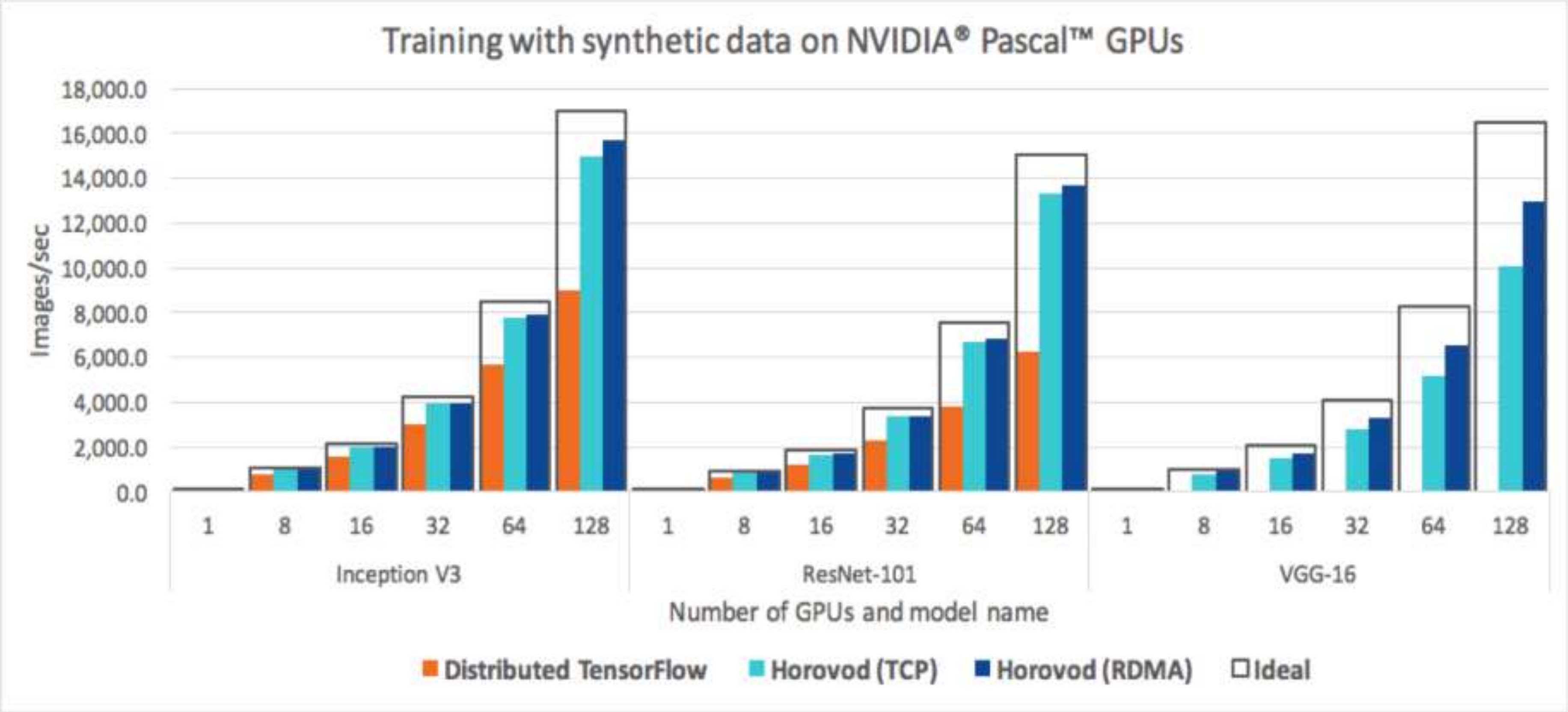
Source Google

Data Parallelism in Practice



Master

Uber Horovod



Benchmark on 32 servers with 4 Pascal GPUs each connected by RoCE-capable 25 Gbit/s network

DELL EMC