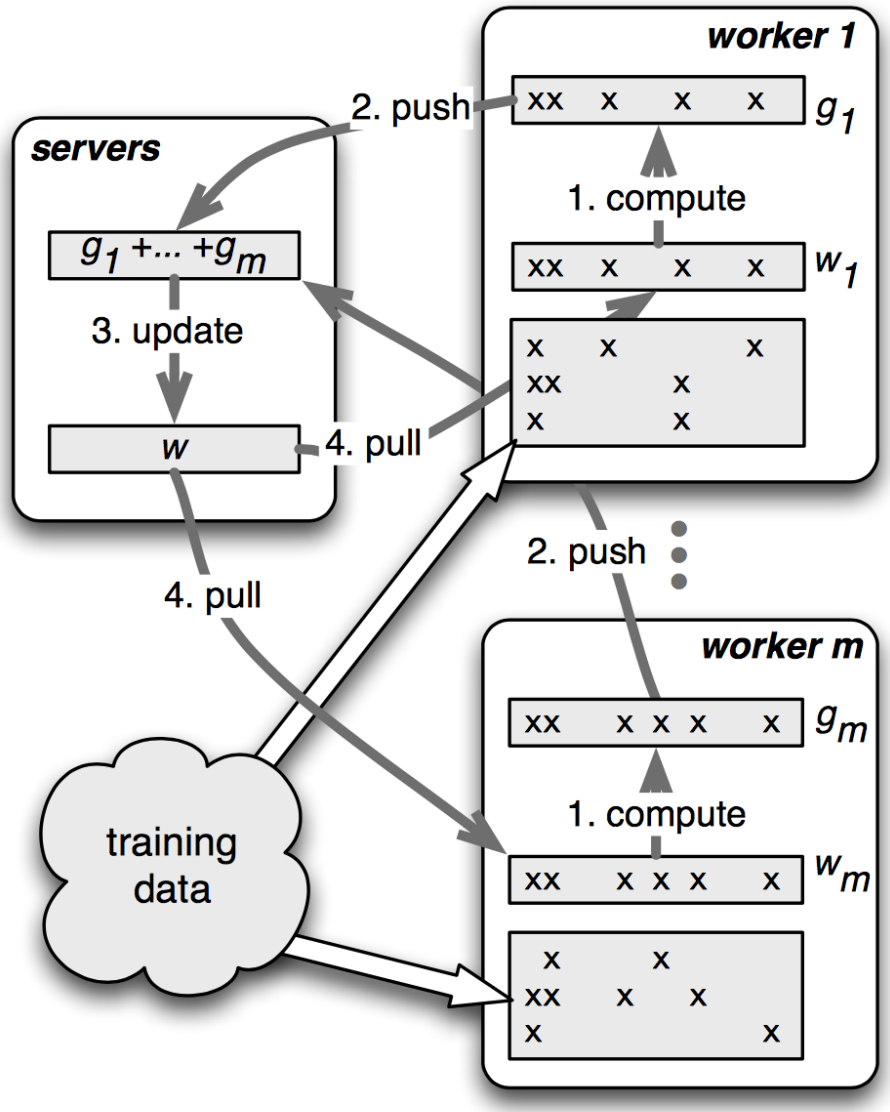
LSML #5

Введение в TensorFlow

Parameter Server (PS)

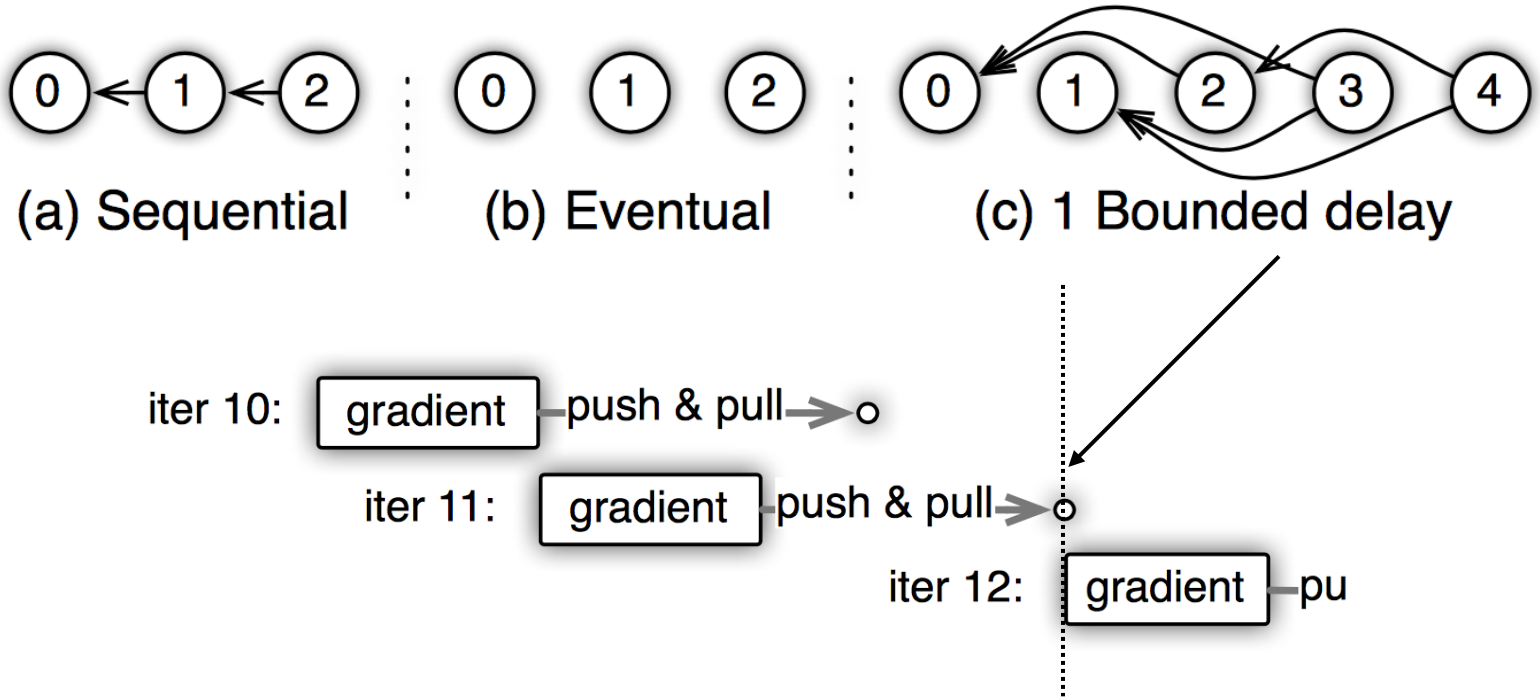


На примере логистической регрессии:

1. Воркеры читают свою часть данных и считают градиенты **g**
2. Воркеры отсылают (**push to PS**) градиенты **g** на сервер параметров
3. Сервер параметров обновляет свои веса **w** на базе сообщений от воркеров
4. Воркеры просят у сервера параметров (**pull from PS**) обновленные веса **w**

Parameter Server (PS)

- AllReduce в VW заточен под сложение, PS – более общий подход
- PS может состоять из многих машин
- Можно варьировать **консистентность** параметров, что позволяет делать вычисления **асинхронно** (например, Async-SGD)



Delayed Block Proximal Gradient – пример с PS

Scheduler:

1: Partition features into b ranges $\mathcal{R}_1, \ldots, \mathcal{R}_b$
2: **for** $t = 0$ **to** T **do**
3: Pick random range $\mathcal{R}_{i_t}$ and issue task to workers
4: **end for**

Разбиваем все признаки на $\boldsymbol{b}$ блоков

Worker r at iteration t

1: Wait until all iterations before $t - \tau$ are finished
2: Compute first-order gradient $g_r^{(t)}$ and diagonal second-order gradient $u_r^{(t)}$ on range $\mathcal{R}_{i_t}$
3: Push $g_r^{(t)}$ and $u_r^{(t)}$ to servers with the KKT filter
4: Pull $w_r^{(t+1)}$ from servers

Не хотим отставать в значении параметра $\boldsymbol{w}$ больше чем на $\boldsymbol{\tau}$ блоков

Servers at iteration t

1: Aggregate gradients to obtain $g^{(t)}$ and $u^{(t)}$
2: Solve the proximal operator

$$w^{(t+1)} \leftarrow \operatorname*{argmin}_u \Omega(u) + \frac{1}{2\eta}\|w^{(t)} - \eta g^{(t)} + u\|_H^2,$$

where $H = \operatorname{diag}(h^{(t)})$ and $\|x\|_H^2 = x^T H x$

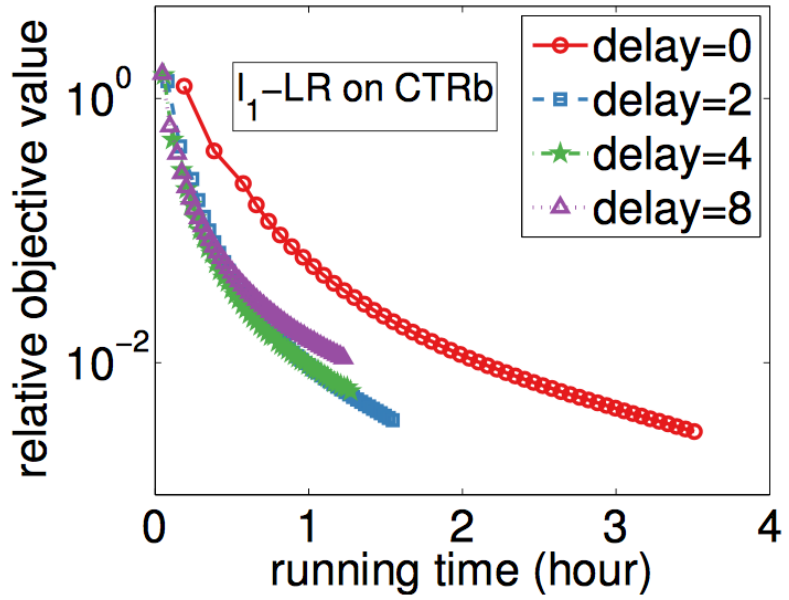
Считается по параметрам итерации $\boldsymbol{t}$, которые у нас есть на этот момент

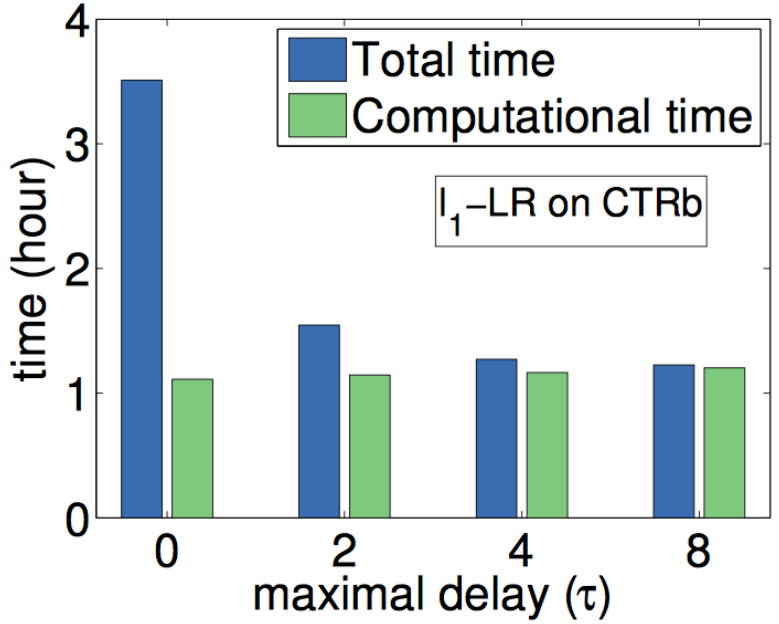
Отсылается воркеру сразу, хранится внутри последнее посчитанное значение $\boldsymbol{w^{t+1}}$

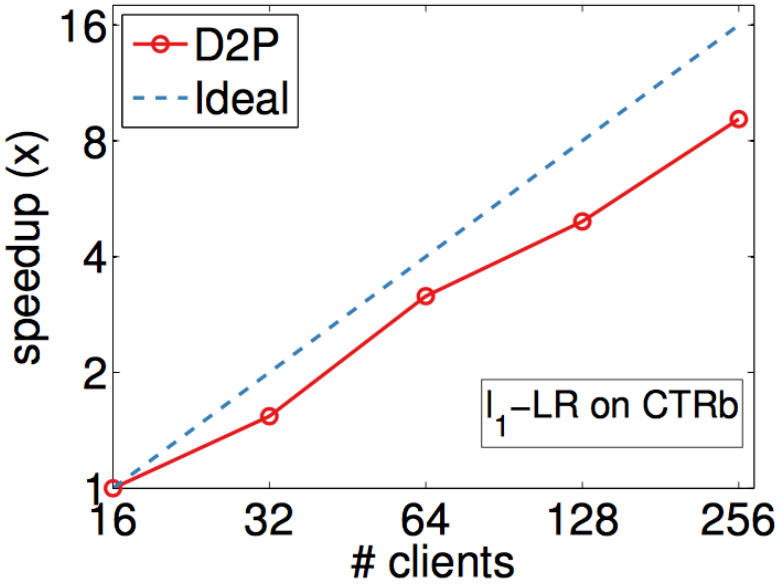
https://www.cs.cmu.edu/~muli/file/parameter_server_osdi14.pdf

Delayed Block Proximal Gradient – пример с PS

- На одной машине сходится медленнее из-за задержек обновления весов
- В распределенном варианте работает быстрее из-за меньшего количества синхронизаций (в τ раз меньше)







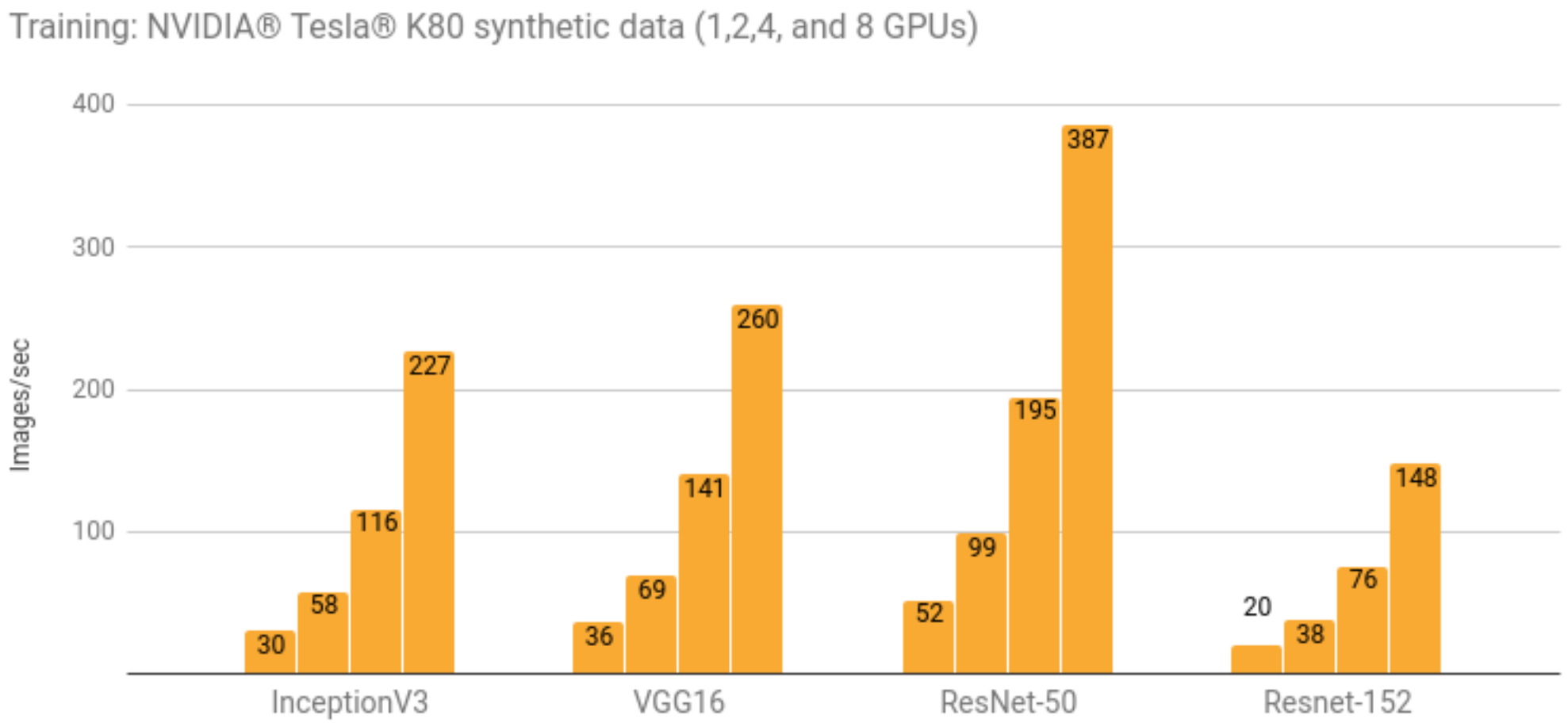
http://opt.kyb.tuebingen.mpg.de/papers/opt2013_submission_1.pdf

TensorFlow (TF)

- Фреймворк для **дифференцируемых вычислений** (у любого узла графа вычислений можно посчитать производную выхода по параметрам)
- Работает на CPU и GPU
- Много готовых операций в библиотеке (полносвязный слой нейросети, LSTM, dropout, функции потерь, функции распределения, …)
- **Символьное** дифференцирование (в Theano можно напечатать **формулу для градиента,** в TF это граф вычислений градиента)
- Умеет работать на сотнях машин с GPU

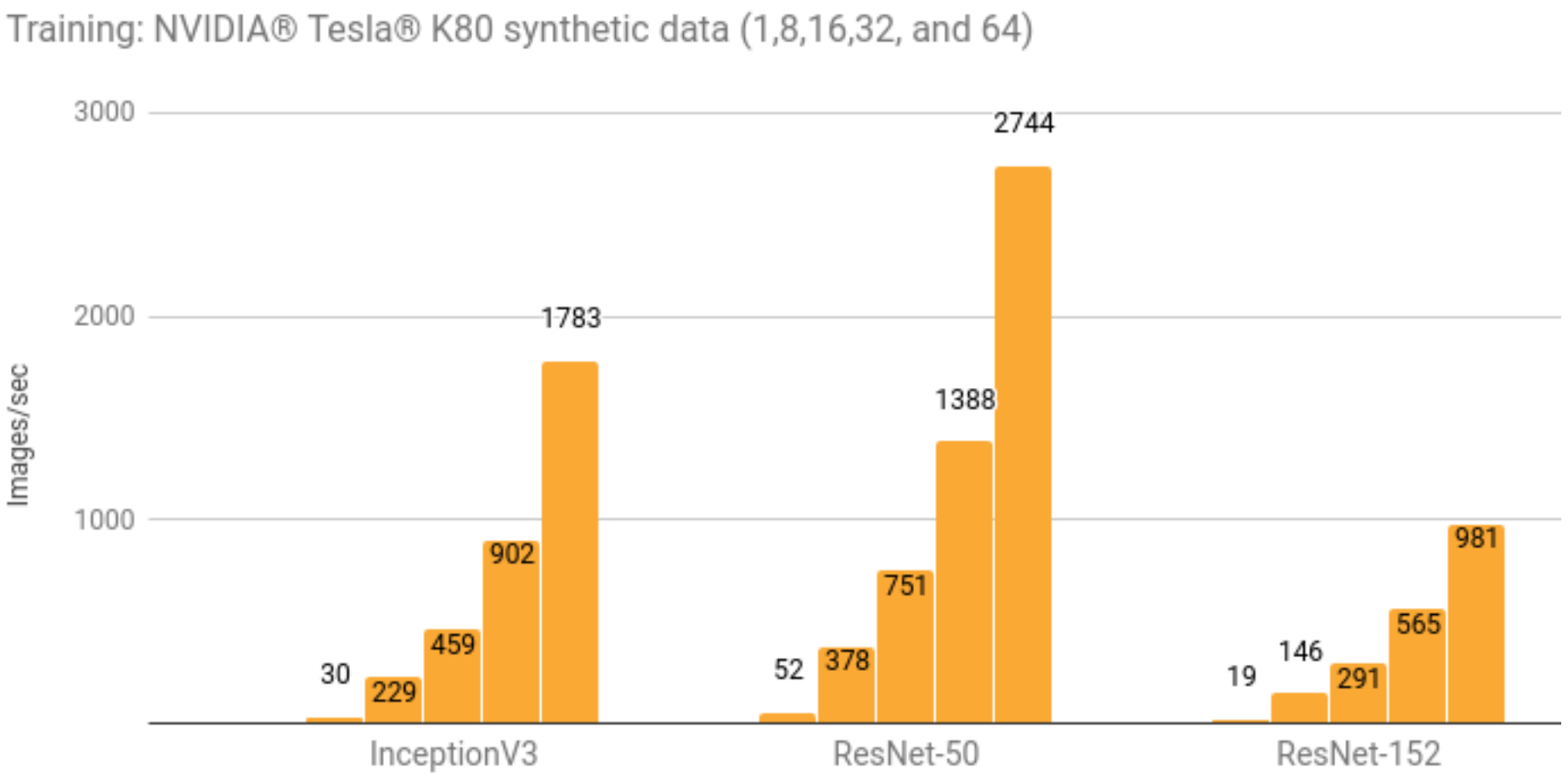
http://deeplearning.net/software/theano/tutorial/gradients.html

TF на одной машине с GPU



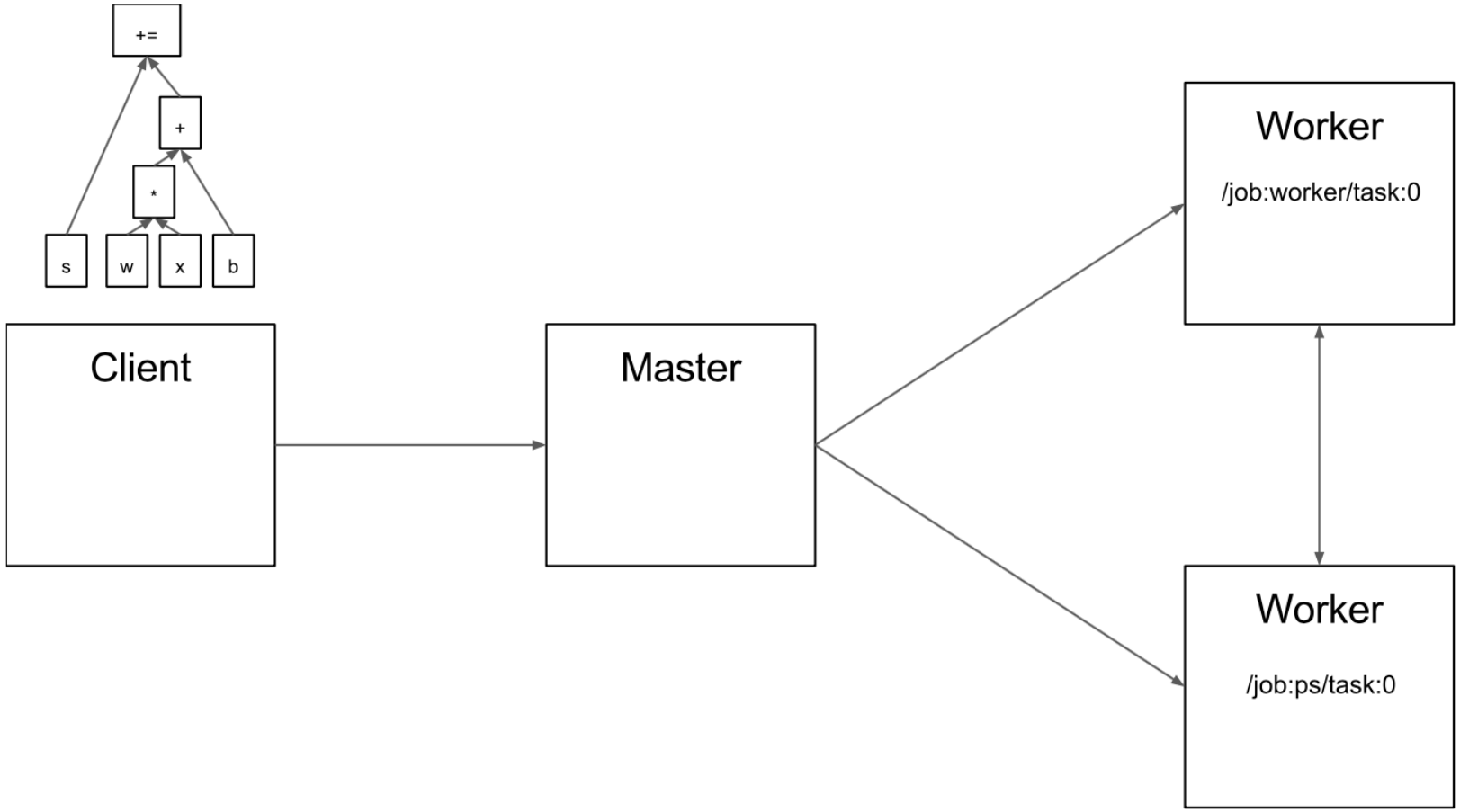
https://www.tensorflow.org/performance/benchmarks

TF на кластере с GPU



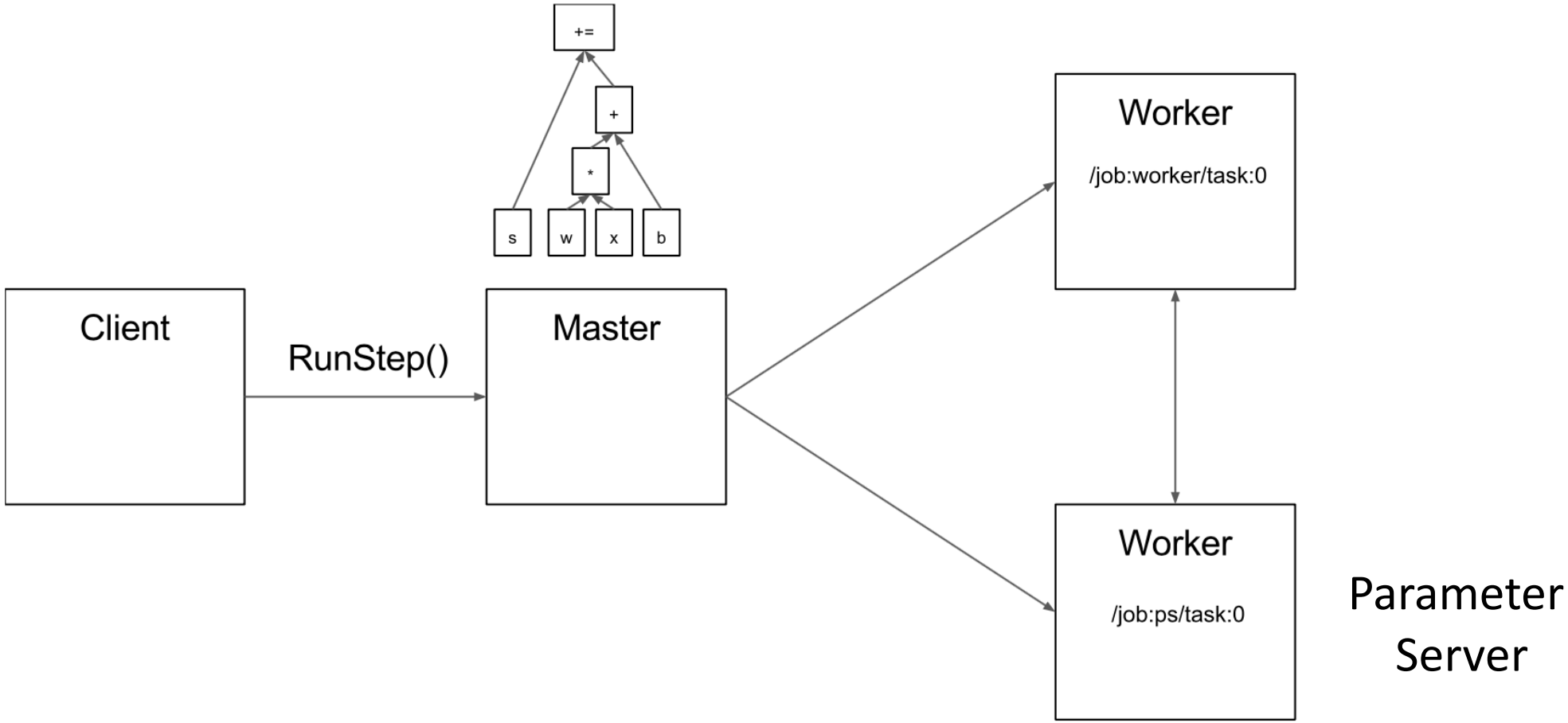
Программа на TF

- На **клиенте** (Python программа) создается **граф вычислений**:



Программа на TF

- Когда просим произвести вычисления, **мастер** (C++) вычисляет граф на **воркерах**



Программа на TF

```
import tensorflow as tf
import numpy as np
```

Импортировали TF

```
trX = np.linspace(-1, 1, 101)
trY = 2 * trX + np.random.randn(*trX.shape) * 0.33
```

Сгенерировали тестовые данные

```
x = tf.placeholder("float")
y = tf.placeholder("float")
w = tf.Variable(0.0, name="weights")

y_pred = tf.multiply(x, w)
loss = tf.square(y - y_pred)
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(loss)
```

Создали переменные для графа

*Наша модель – x * w*

Функция потерь

Шаг по градиенту loss

```
with tf.Session() as sess:
    tf.global_variables_initializer().run()
    for i in range(10):
        for (_x, _y) in zip(trX, trY):
            sess.run(train_step, feed_dict={x: _x, y: _y})
    print(sess.run(w))
```

Создали сессию (создает граф на мастере)

Инициализировали переменные

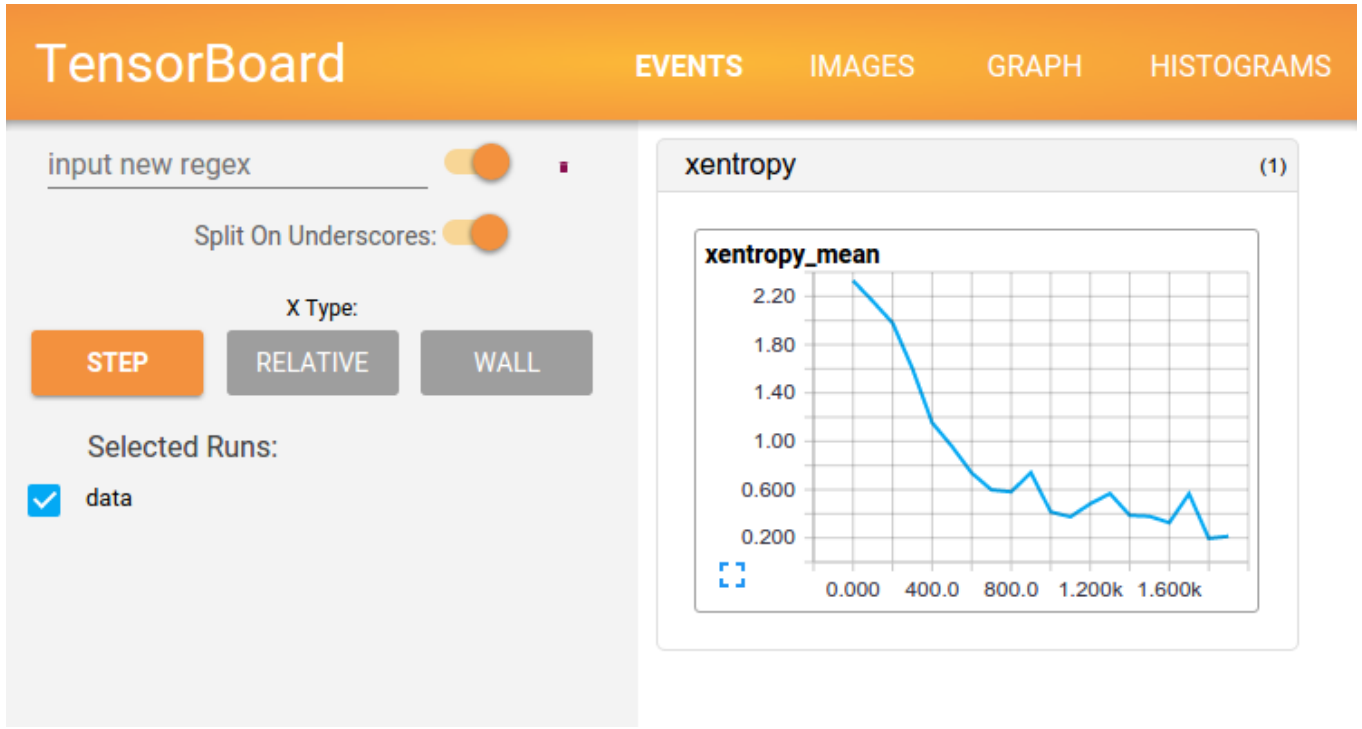
Делаем шаг по градиенту loss

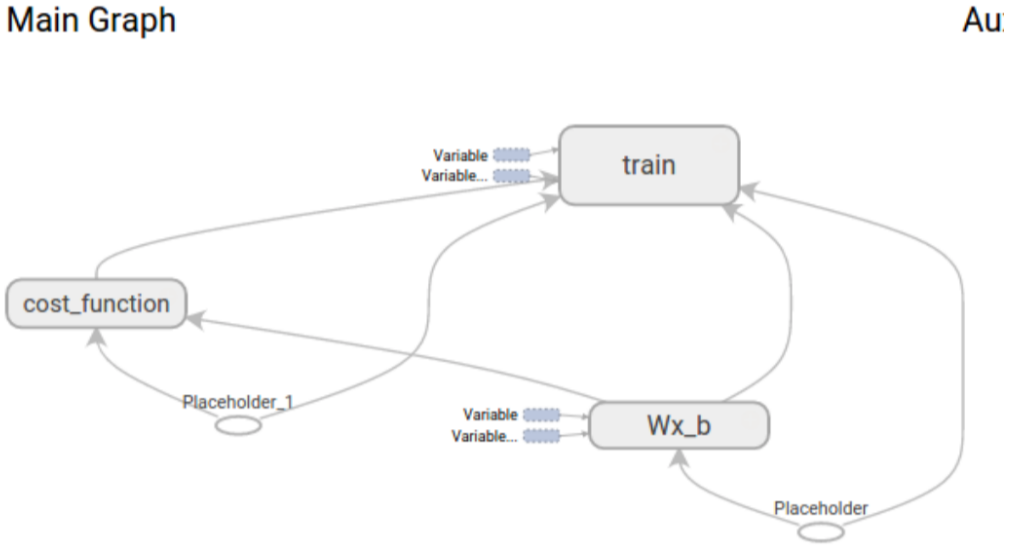
Получаем результирующий w

```
2.04825
```

TensorBoard

- Визуализация статистик узлов графа, имбеддингов, графа вычислений





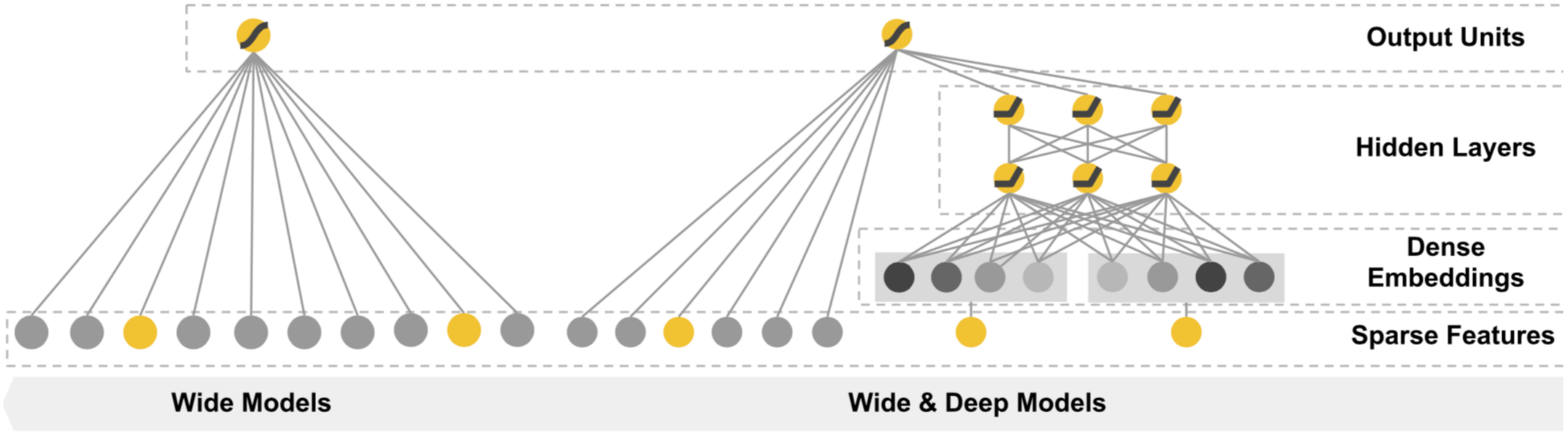
https://www.tensorflow.org/get_started/summaries_and_tensorboard

Какие задачи можно решать в TF

- Анализ картинок (CNN)
- Звука (RNN)
- Текстов (RNN)
- …

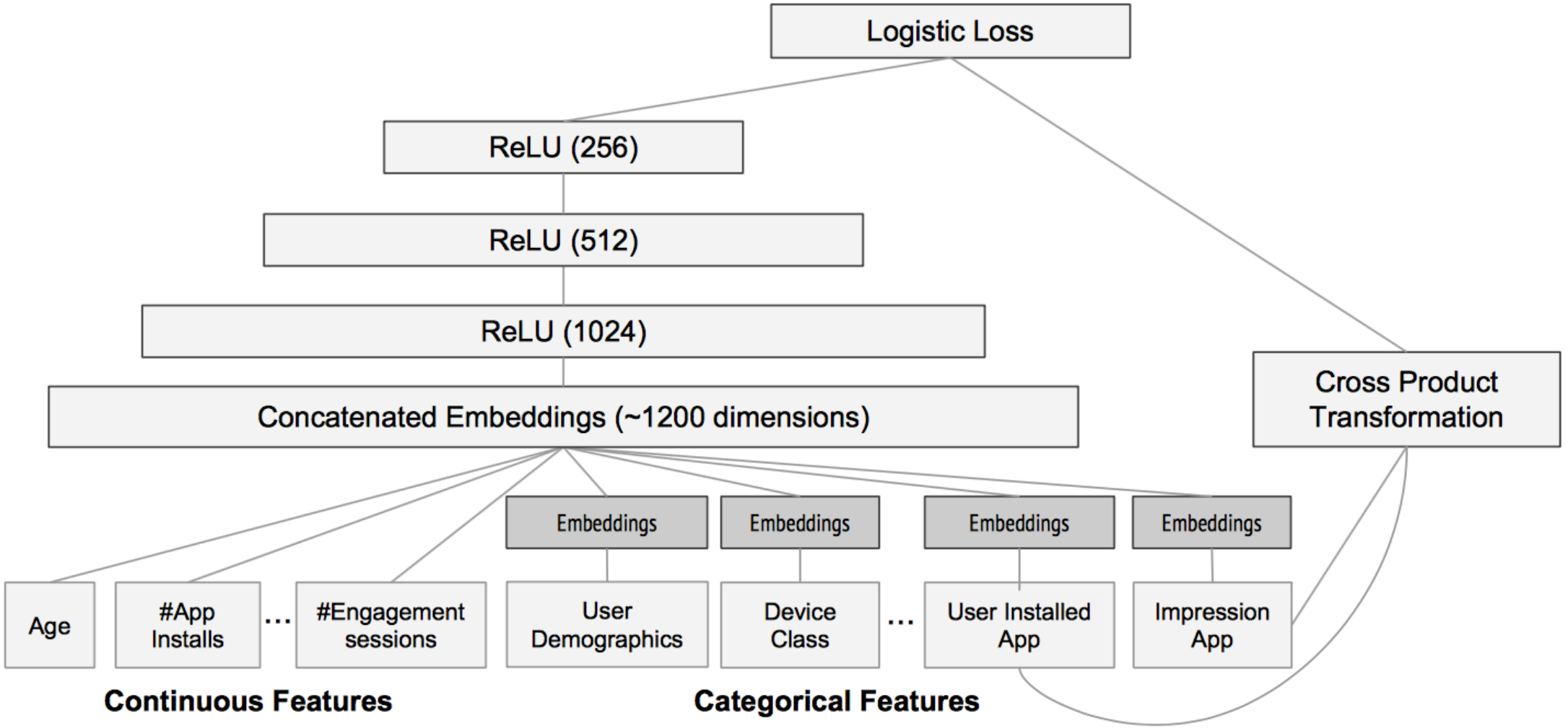
И другие задачи ML

- Линейные модели
- Wide & deep модели (имбеддинги категориальных признаков)



https://www.tensorflow.org/tutorials/wide_and_deep

Wide & deep для рекомендаций в Play Store

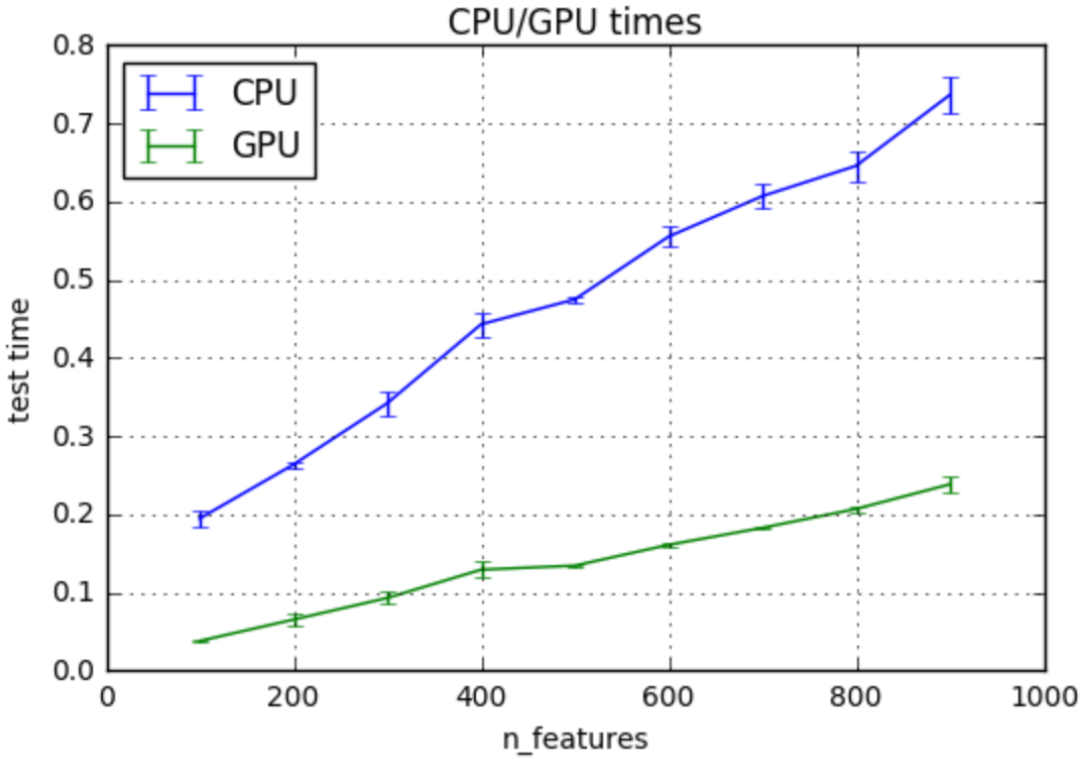


https://arxiv.org/pdf/1606.07792.pdf

Factorization Machines

- Есть реализация FM (с SGD) на TF

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} x_j x_{j'} \sum_{f=1}^{k} v_{j,f}\, v_{j',f}$$



http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.442.5724&rep=rep1&type=pdf
https://github.com/geffy/tffm/blob/master/gpu_benchmark.ipynb

Keras – библиотека поверх TF

```
import tensorflow as tf
import tensorflow.contrib.keras as keras
import numpy as np
```

```
trX = np.linspace(-1, 1, 101)
trY = 2 * trX + np.random.randn(*trX.shape) * 0.33
```

```
model = keras.models.Sequential()
model.add(keras.layers.Dense(1, input_shape=(1,)))
model.compile(optimizer='sgd', loss='mse', metrics=['mse'])
```

```
model.fit(trX, trY, batch_size=1, epochs=10, verbose=0)
model.get_weights()
```

```
[array([[ 2.11763072]], dtype=float32), array([-0.02399813], dtype=float3
2)]
```

Ссылки

- Статья о Parameter Server
  https://www.cs.cmu.edu/~muli/file/parameter_server_osdi14.pdf
- Еще про PS
  http://opt.kyb.tuebingen.mpg.de/papers/opt2013_submission_1.pdf
- TensorFlow tutorials https://www.tensorflow.org/tutorials
- Можно писать свои операции на C++
  https://github.com/kopopt/fast_tffm/blob/master/cc/fm_grad_op.cc