LSML #7

Распараллеливание нейросетей

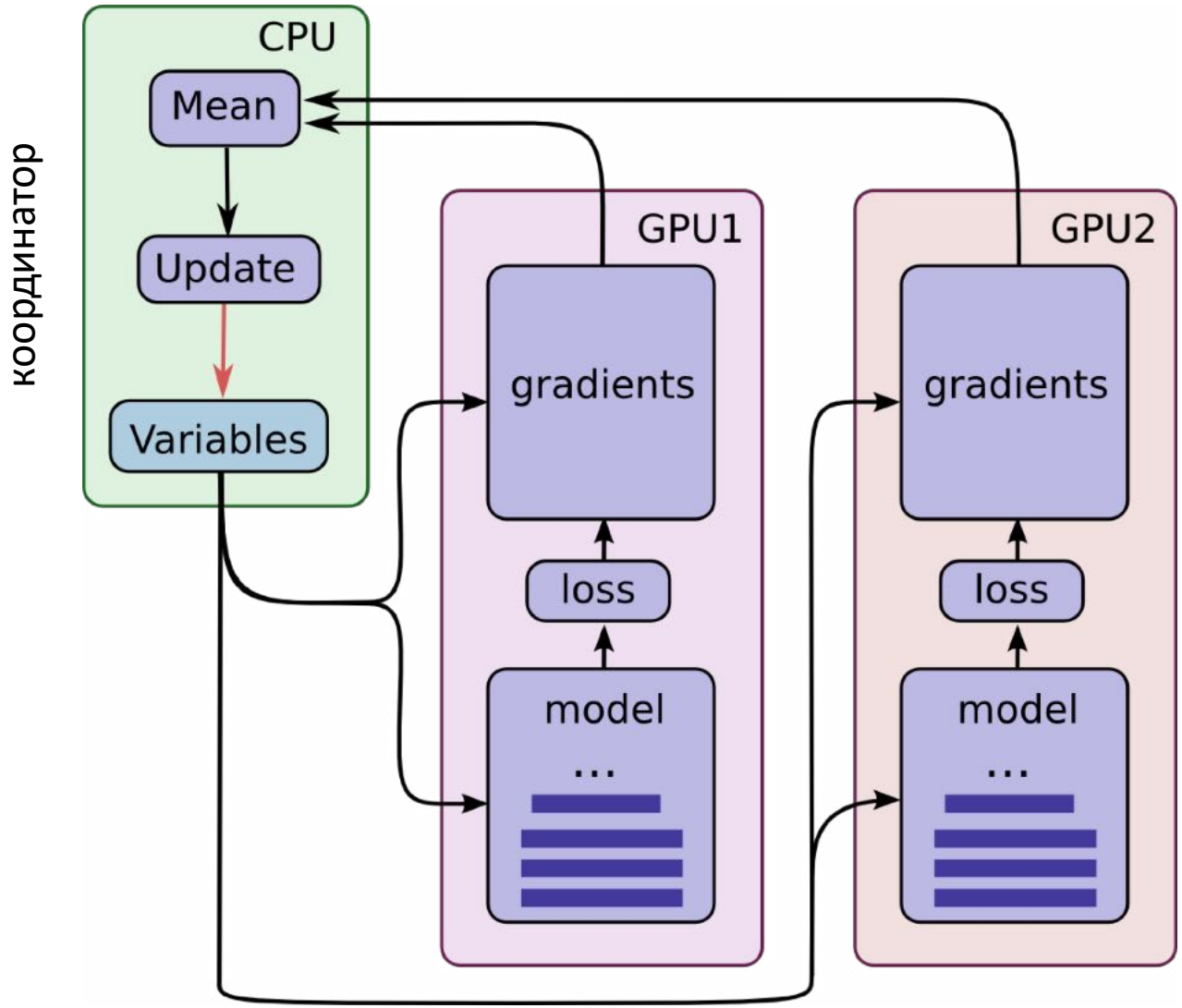
Как ускорять применение нейросети

- Если предсказываем для 1 примера:
  - Используют быстрый GPU (+ TensorRT) https://venturebeat.com/2018/03/27/nvidia-speeds-up-deep-learning-inference-processing/
  - Сжимают сети (teacher networks) https://arxiv.org/pdf/1412.6550.pdf
  - Квантуют веса сетей (быстрые INT8 операции) https://www.tensorflow.org/performance/quantization https://www.theregister.co.uk/2016/09/13/nvidia_p4_p40_gpu_ai/
  - Придумывают архитектуры для мобильных (MobileNet) https://arxiv.org/pdf/1704.04861.pdf
  - …
- Если предсказываем для кучи примеров:
  - Embarrassingly parallel (предсказания делаются независимо)

Будем параллелить обучение

- Есть два подхода:
  - Data-parallel (по данным)
  - Model-parallel (по весам)

Data-parallel



- Рассылаем копию модели на каждый GPU
- Применяем независимо к разным кусочкам батча
- Вклады в градиент отправляем **координатору** на CPU (или GPU), суммируем и заново рассылаем всем GPU

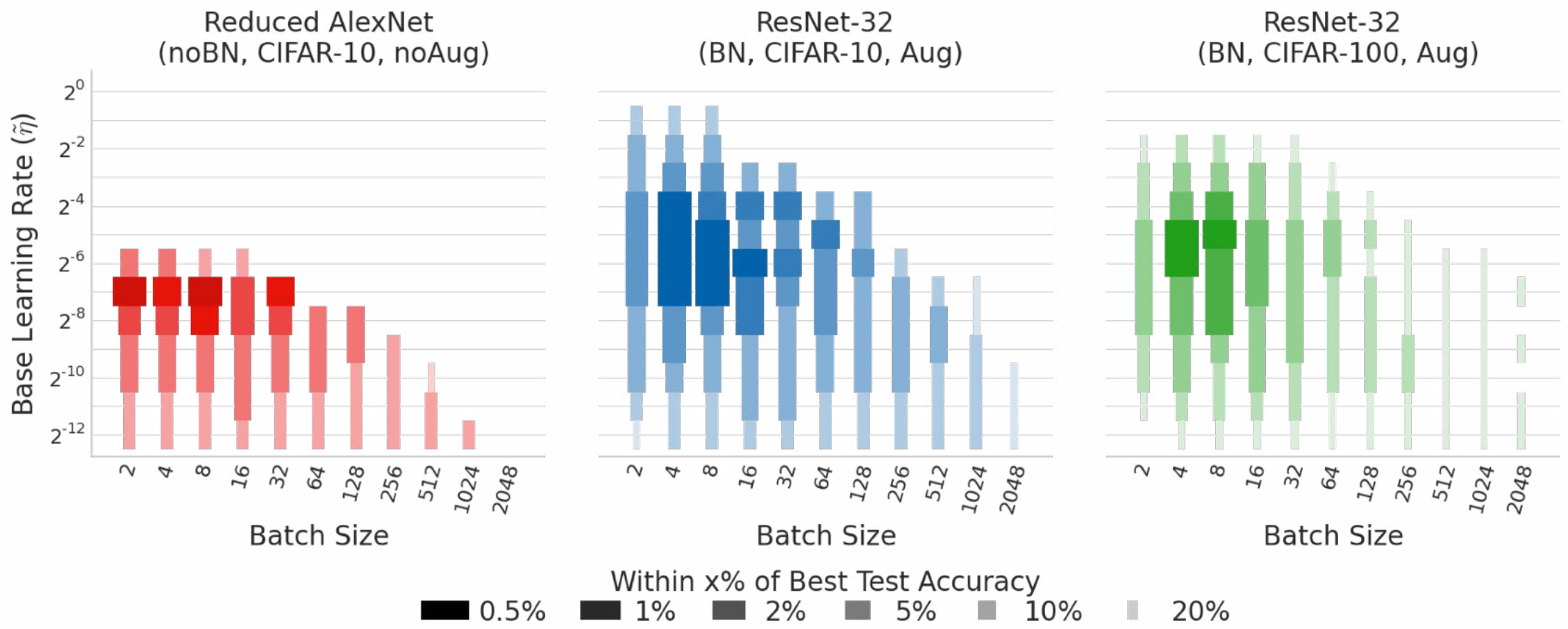
https://github.com/awslabs/deeplearning-benchmark/tree/master/tensorflow/inception

Data-parallel

- **Все упирается в размер батча, сойдется ли с большим?**
- Если модель большая, то долгие пересылки

Размер батча

- Кто-то говорит, что нужно брать маленький:



https://arxiv.org/pdf/1609.04836.pdf https://arxiv.org/pdf/1804.07612.pdf https://www.graphcore.ai/posts/revisiting-small-batch-training-for-deep-neural-networks

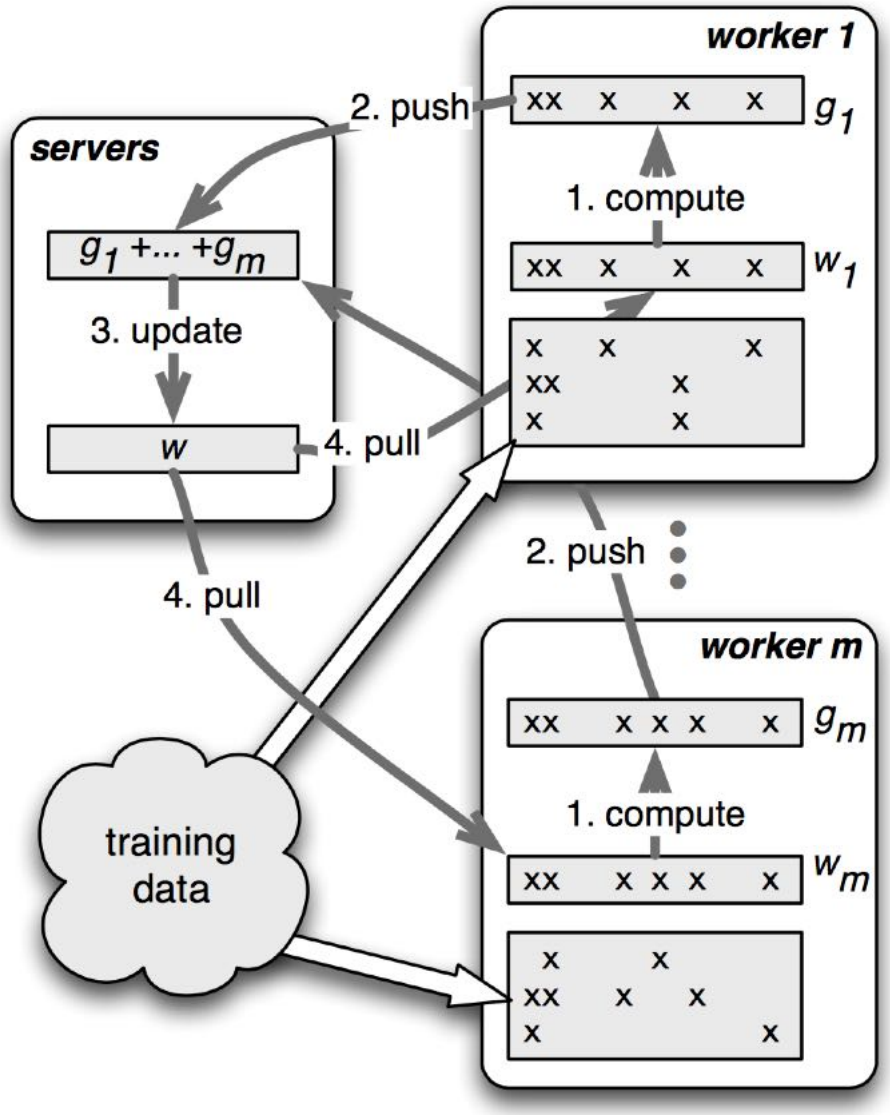
Размер батча

- Маленький плох для распараллеливания:
  - Не полностью утилизируется GPU
  - Маленькое количество воркеров

Data-parallel

- Все упирается в размер батча, сойдется ли с большим?
- **Если модель большая, то долгие пересылки**

Усложним координатор: Parameter Server (PS)



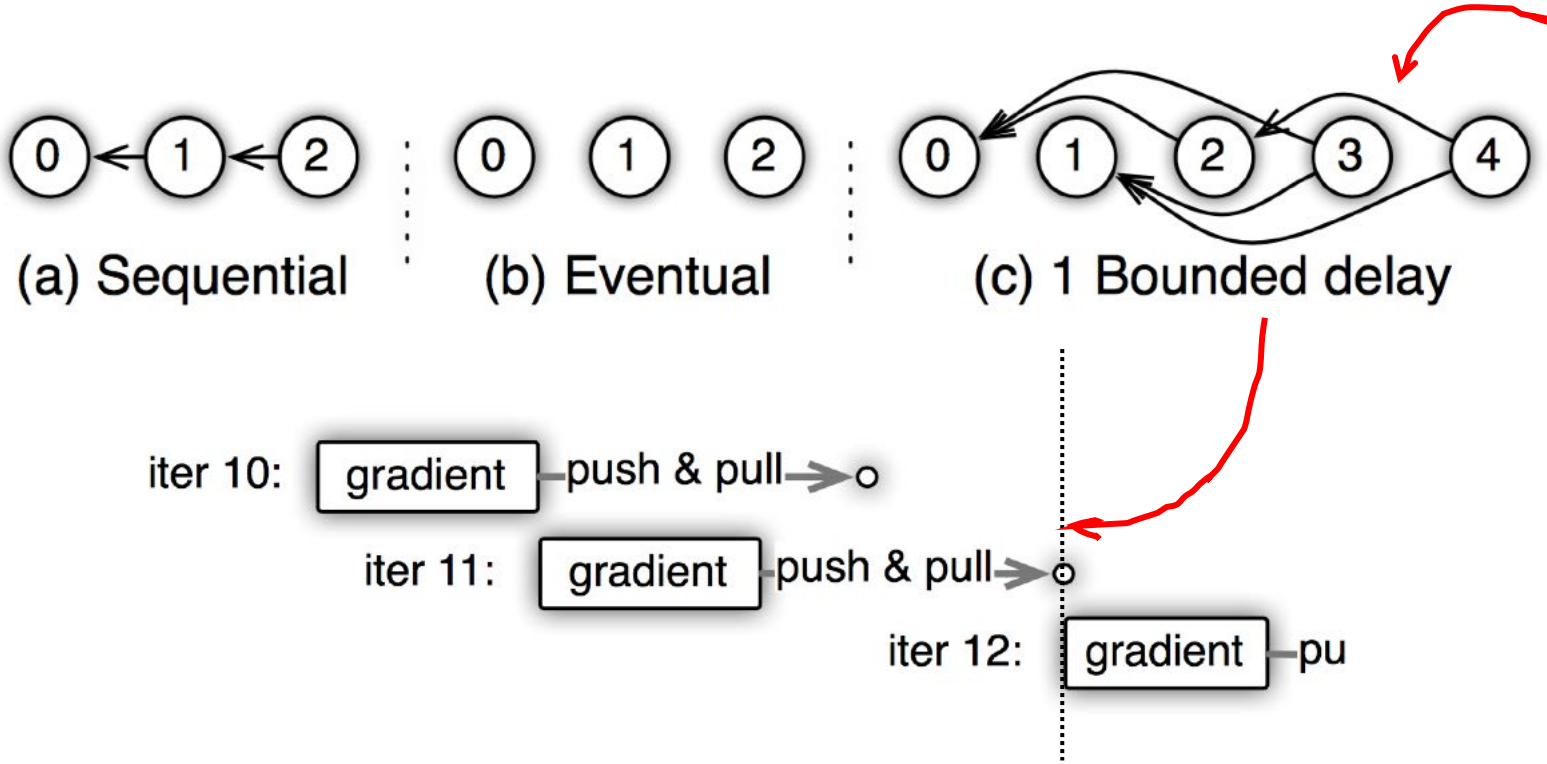
На примере линейной регрессии:

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**

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

Parameter Server (PS)

- PS может состоять из многих машин (шардирование параметров)
- Можно варьировать **консистентность** параметров, что позволяет делать вычисления **асинхронно**



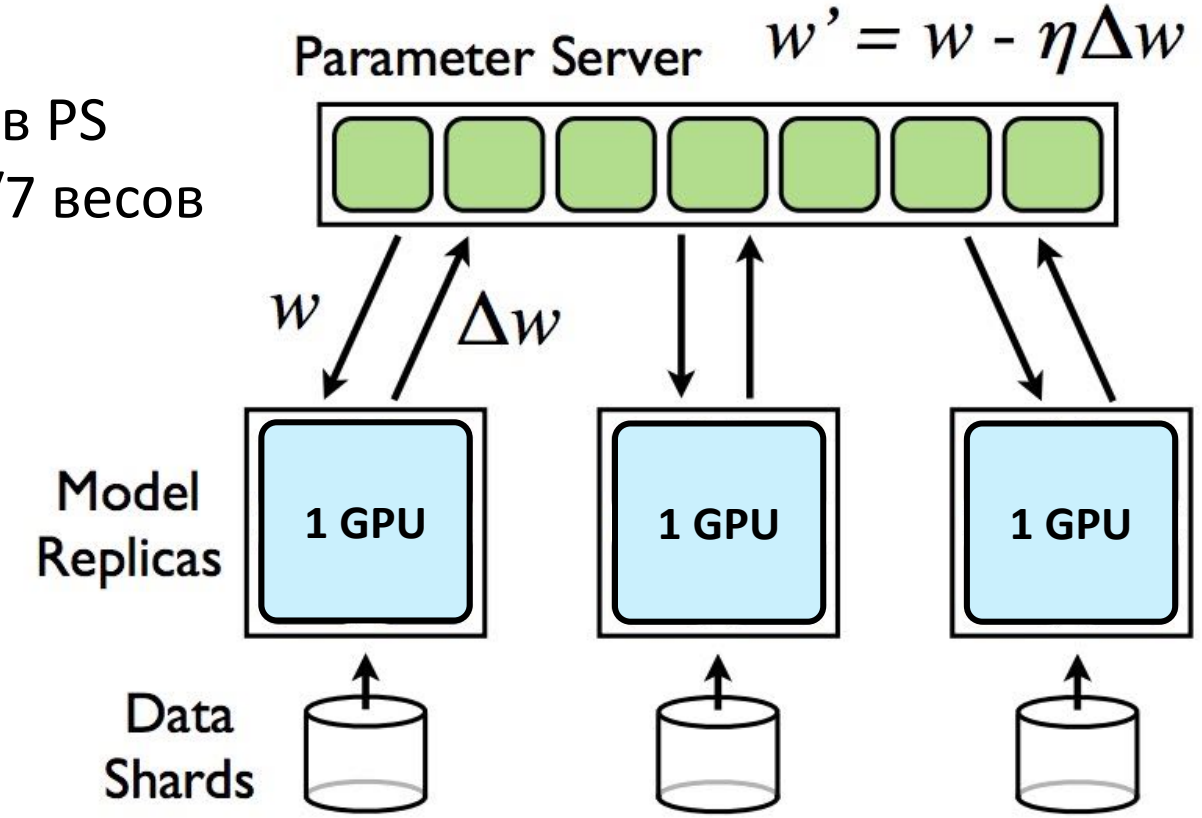
Пример: отставать не больше, чем на 2 апдейта весов

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

Уменьшим overhead от пересылок

- **Асинхронный SGD**: не будем ждать, пока прилетит вклад в градиент от всех машин, сделаем шаг с тем, что есть (возможно только локальный вклад)

каждая машина в PS обрабатывает 1/7 весов



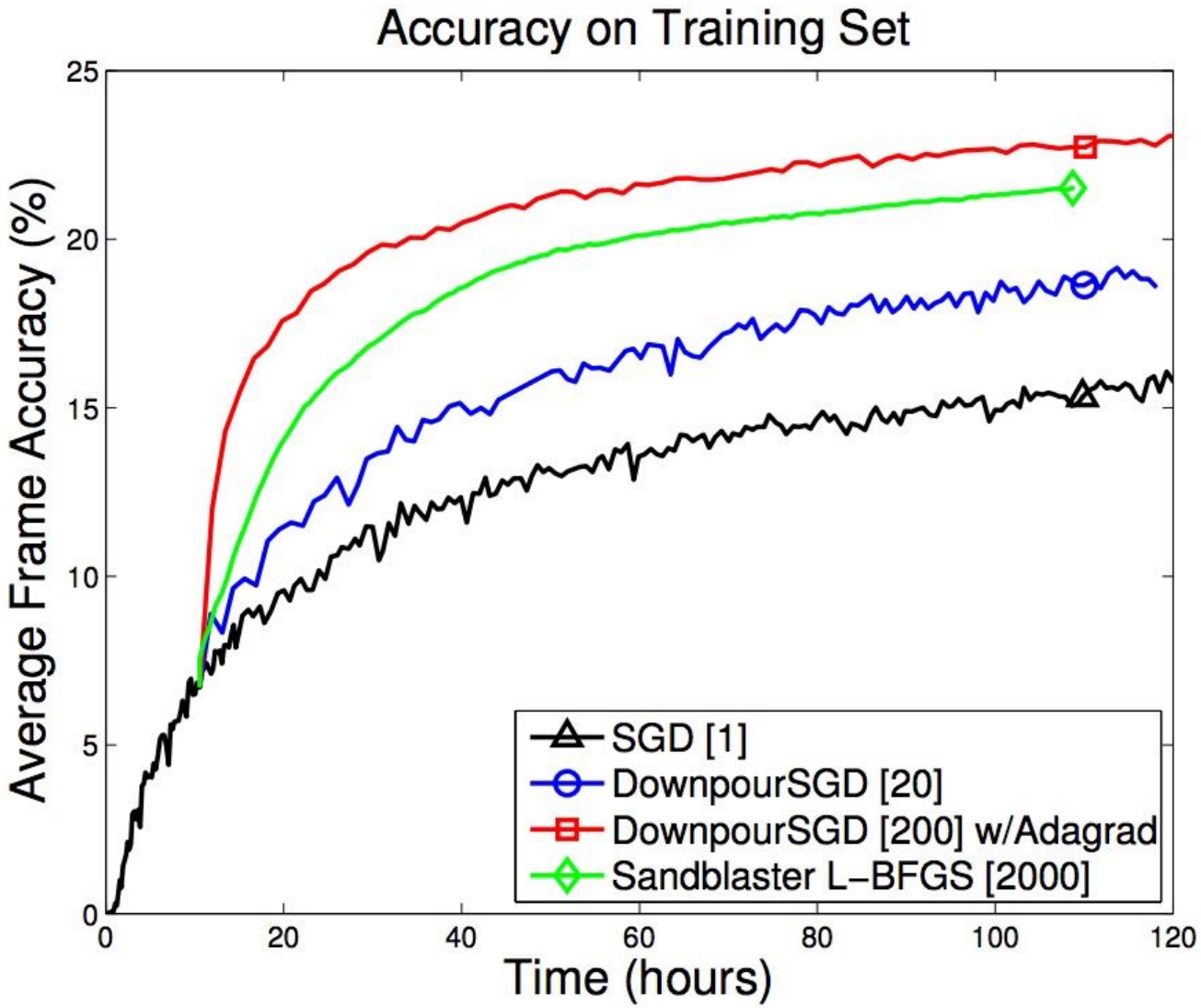
логика обновления параметров в сервере

асинхронно

data-parallel

https://storage.googleapis.com/pub-tools-public-publication-data/pdf/40565.pdf

Асинхронный SGD от Google (DistBelief)



Запускают асинхронное обучение после 1 эпохи, видимо иначе ничего не сходится из-за запаздываний градиентов на ранних стадиях обучения

https://storage.googleapis.com/pub-tools-public-publication-data/pdf/40565.pdf

Delayed Block Proximal Gradient – пример с PS

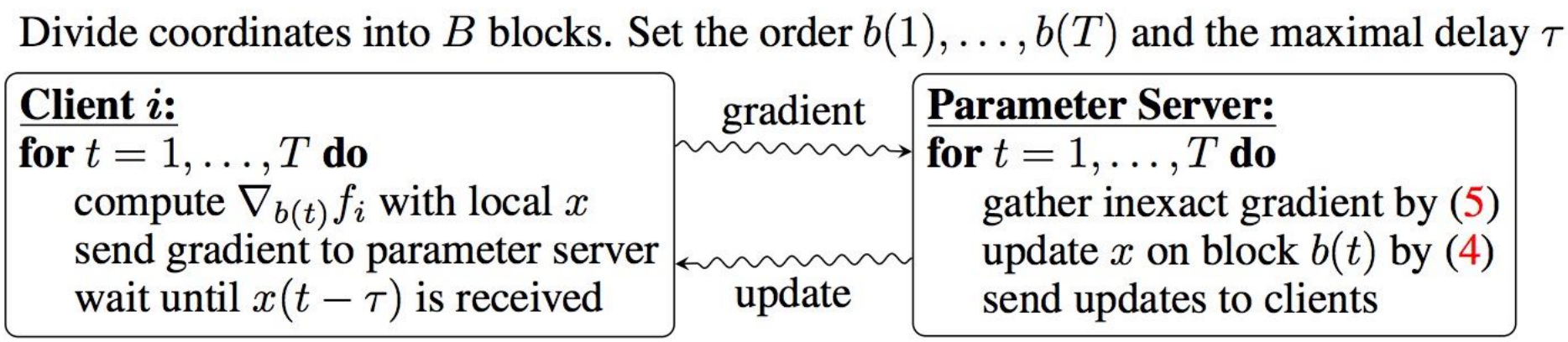


Figure 1: D2P, Distributed Delayed Proximal Gradient Methods. Both clients and the parameter server span several machines. All data sending and receiving are non-blocking.

- Линейная регрессия
- На итерации t обновляем веса в блоке $b(t)$
- Хотим на каждом воркере отставать не больше, чем на τ блоков
- Асинхронно посылаем апдейты весов

http://www.cs.cmu.edu/afs/cs/user/muli/www/file/ddp.pdf

Delayed Block Proximal Gradient – пример с PS

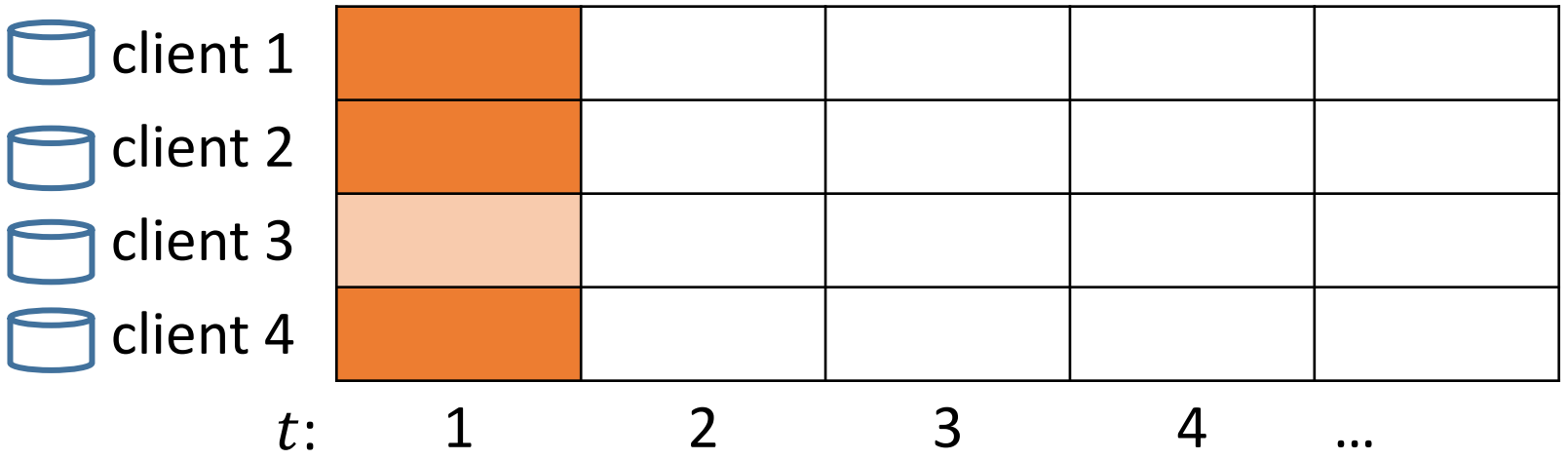
Divide coordinates into B blocks. Set the order $b(1), \ldots, b(T)$ and the maximal delay τ

Client i:
for $t = 1, \ldots, T$ **do**
compute $\nabla_{b(t)} f_i$ with local x
send gradient to parameter server
wait until $x(t - \tau)$ is received

gradient →
← update

Parameter Server:
for $t = 1, \ldots, T$ **do**
gather inexact gradient by (5)
update x on block $b(t)$ by (4)
send updates to clients

Figure 1: D2P, Distributed Delayed Proximal Gradient Methods. Both clients and the parameter server span several machines. All data sending and receiving are non-blocking.



К какому-то моменту первую итерацию завершили 1, 2 и 4 воркеры

http://www.cs.cmu.edu/afs/cs/user/muli/www/file/ddp.pdf

Delayed Block Proximal Gradient – пример с PS

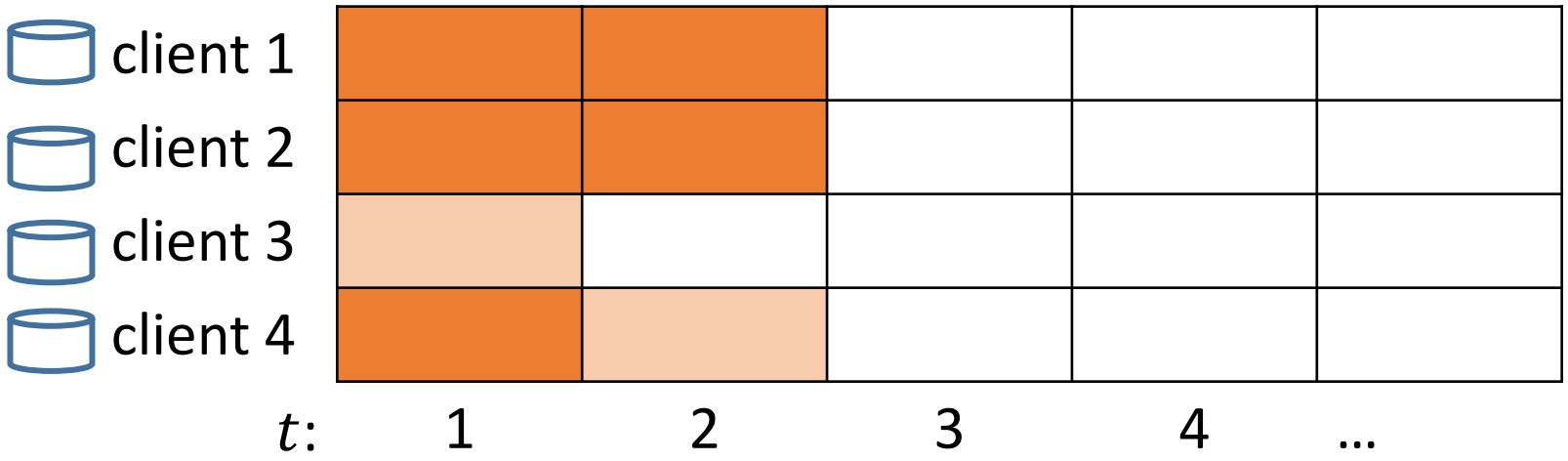
Divide coordinates into B blocks. Set the order $b(1), \ldots, b(T)$ and the maximal delay τ

Client i:
for $t = 1, \ldots, T$ **do**
compute $\nabla_{b(t)} f_i$ with local x
send gradient to parameter server
wait until $x(t - \tau)$ is received

gradient →
← update

Parameter Server:
for $t = 1, \ldots, T$ **do**
gather inexact gradient by (5)
update x on block $b(t)$ by (4)
send updates to clients

Figure 1: D2P, Distributed Delayed Proximal Gradient Methods. Both clients and the parameter server span several machines. All data sending and receiving are non-blocking.



Теперь вторую итерацию закончили 1 и 2 воркеры

http://www.cs.cmu.edu/afs/cs/user/muli/www/file/ddp.pdf

Delayed Block Proximal Gradient – пример с PS

Divide coordinates into B blocks. Set the order $b(1), \ldots, b(T)$ and the maximal delay τ

Client i:
for $t = 1, \ldots, T$ **do**
compute $\nabla_{b(t)} f_i$ with local x
send gradient to parameter server
wait until $x(t - \tau)$ is received

gradient →
← update

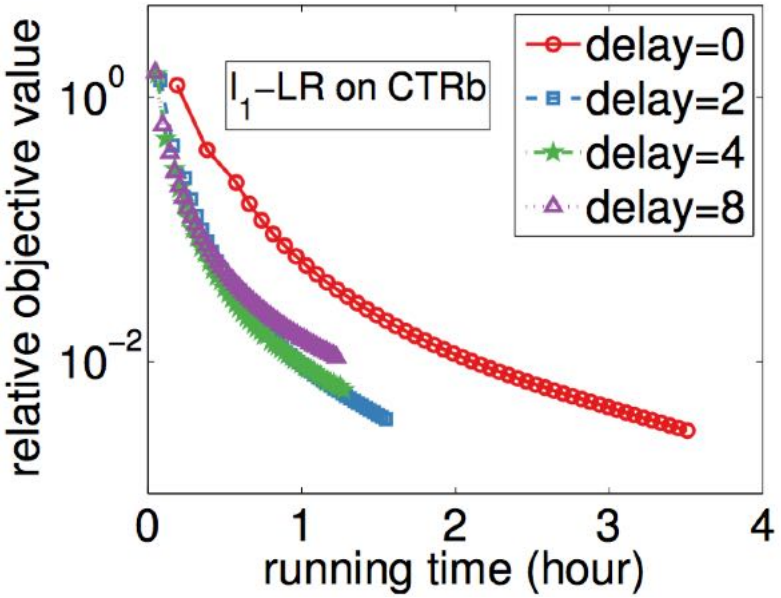
Parameter Server:
for $t = 1, \ldots, T$ **do**
gather inexact gradient by (5)
update x on block $b(t)$ by (4)
send updates to clients

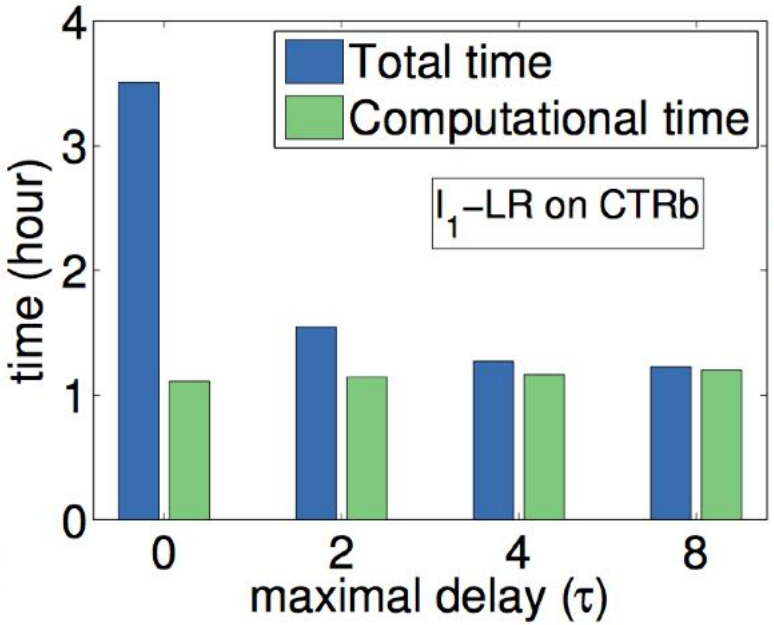
Figure 1: D2P, Distributed Delayed Proximal Gradient Methods. Both clients and the parameter server span several machines. All data sending and receiving are non-blocking.

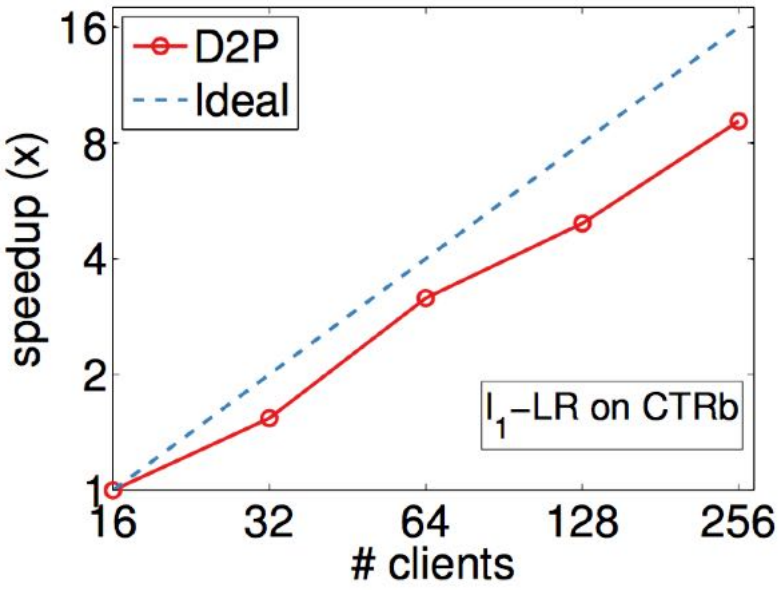
При $\tau = 2$ мы не можем продолжить вычисления на 1 и 2 воркере

Delayed Block Proximal Gradient – пример с PS

- В τ раз меньше синхронизаций
- Замедление сходимости **компенсируется** ускорением одного шага



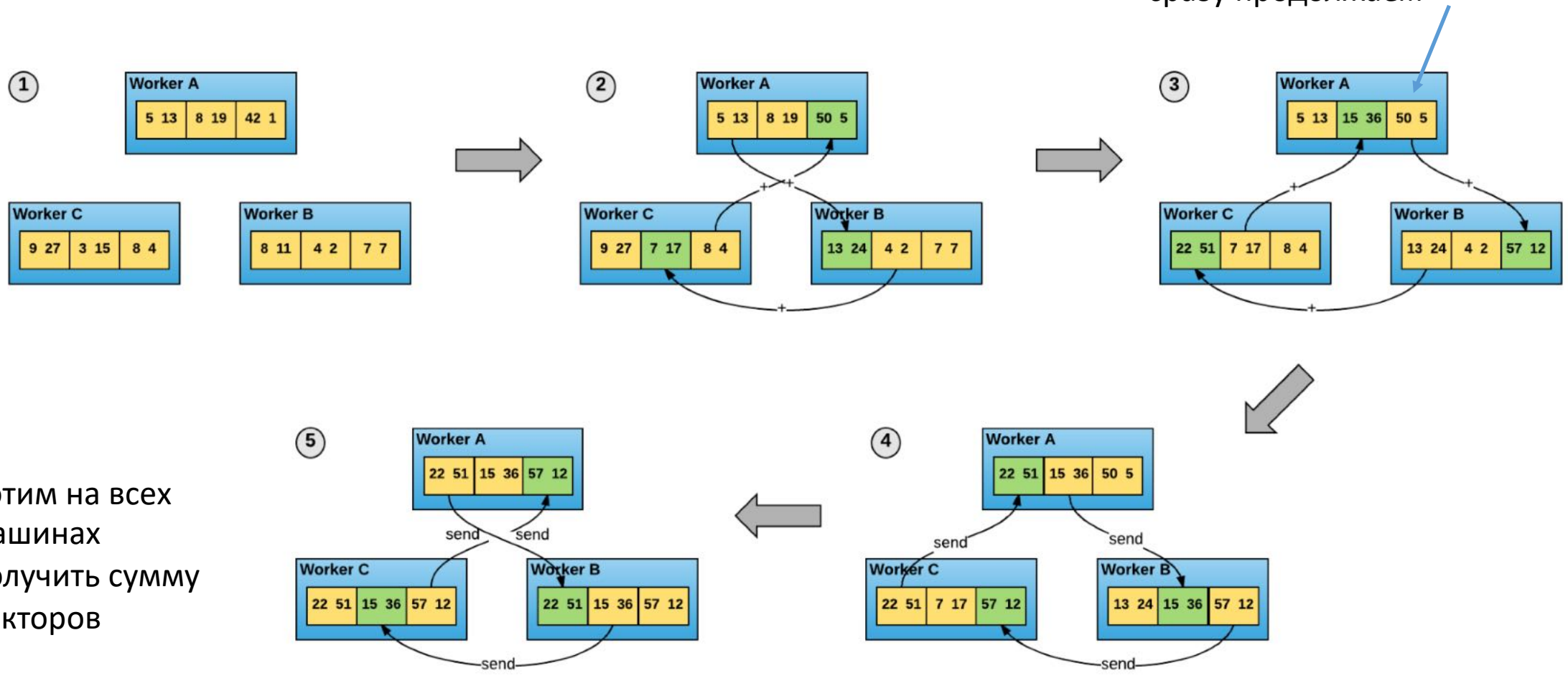




http://opt.kyb.tuebingen.mpg.de/papers/opt2013_submission_1.pdf

Синхронный SGD: Horovod

Как только посчитается на предыдущем шаге, сразу продолжаем



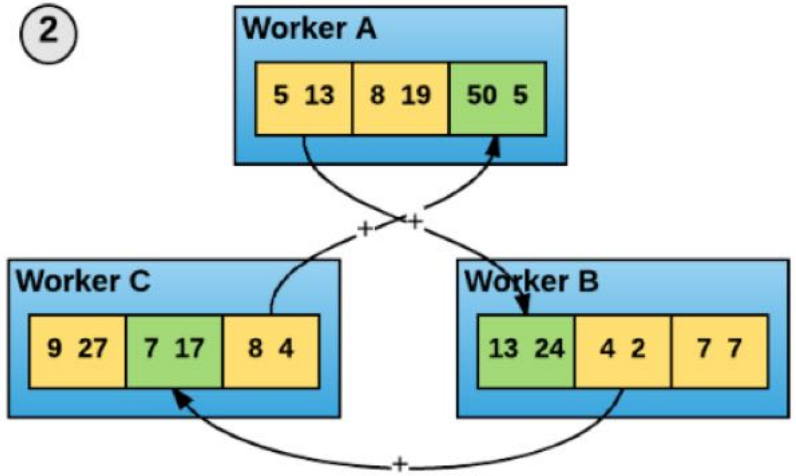
Хотим на всех машинах получить сумму векторов

Чем-то похоже на AllReduce в VW, только без дерева

https://eng.uber.com/horovod/

Синхронный SGD: Horovod

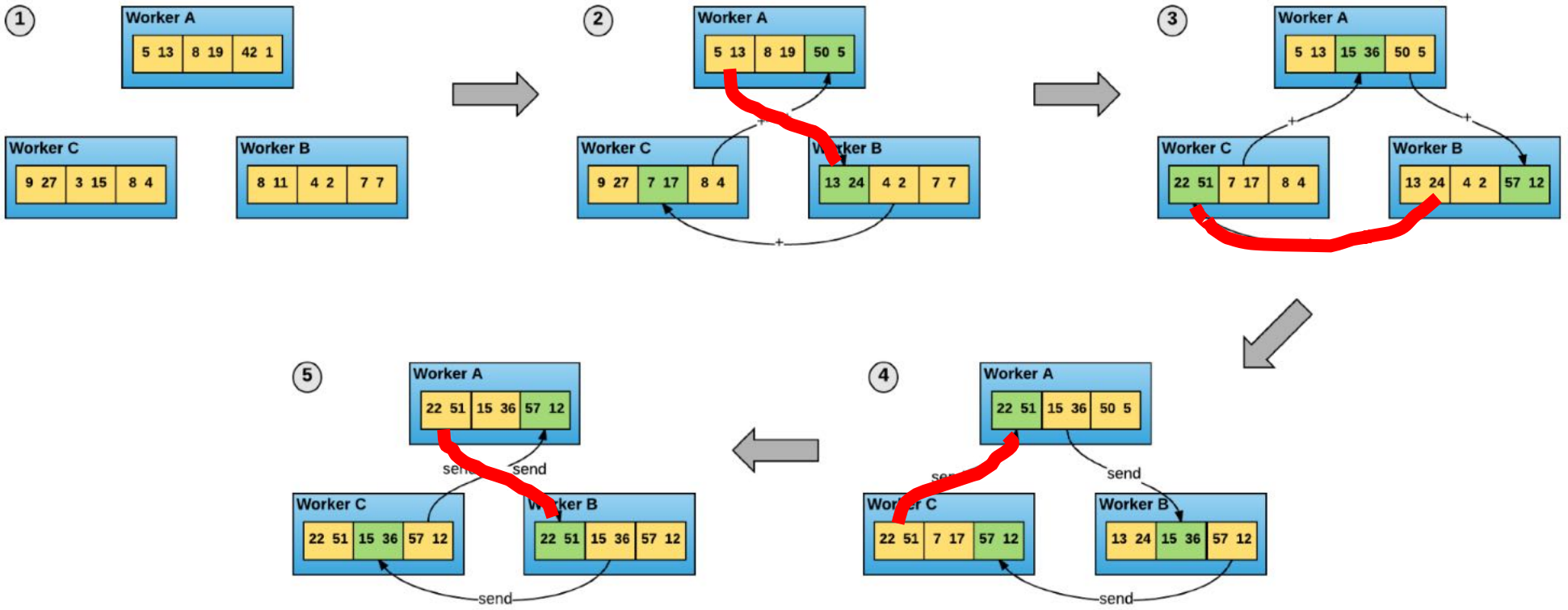
- Data-parallel (режем batch)
- Все пересылки peer-to-peer
- Вот тут нет блокировок (эффективно утилизирует сеть и GPU):



https://eng.uber.com/horovod/

Синхронный SGD: Horovod

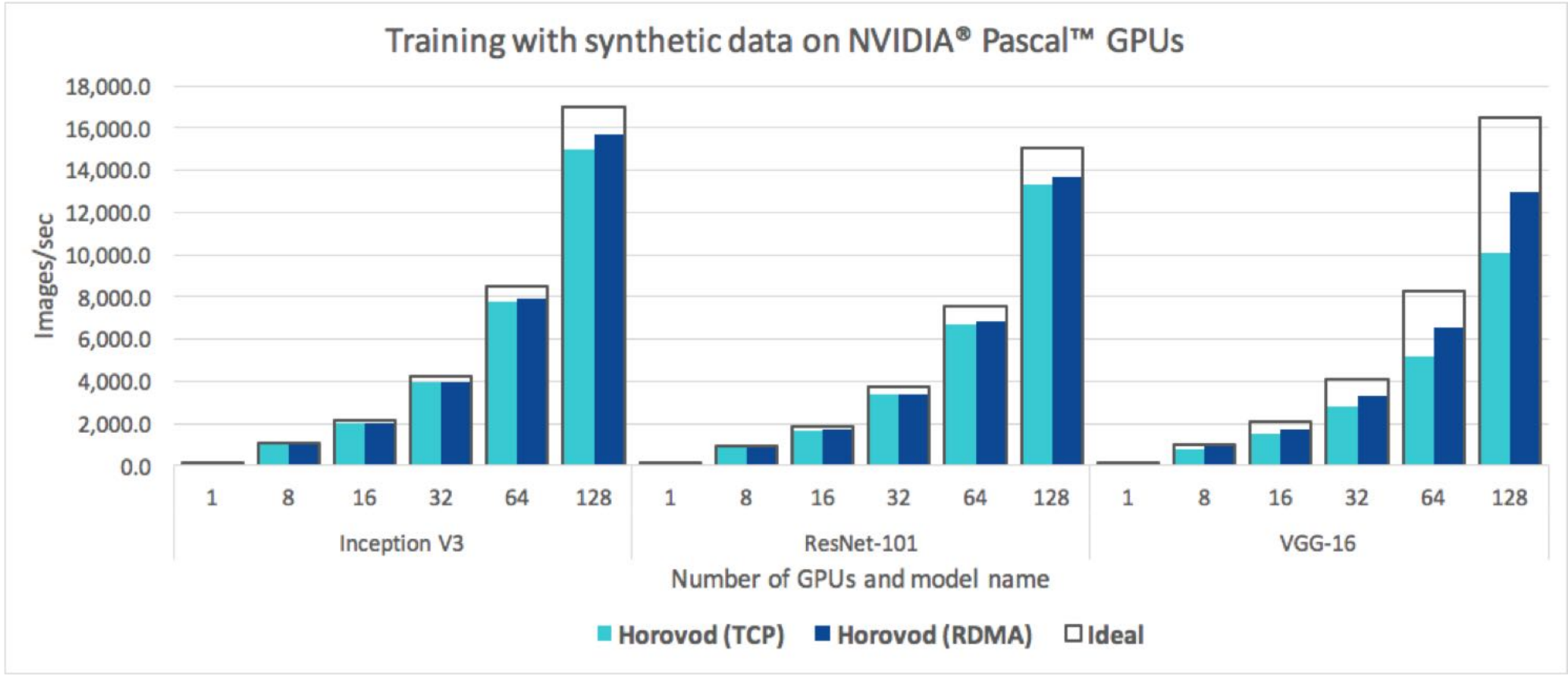
- По сути мы обмениваемся каждым блоком (независимо) по кругу (видимо поэтому так и назвали):



https://eng.uber.com/horovod/

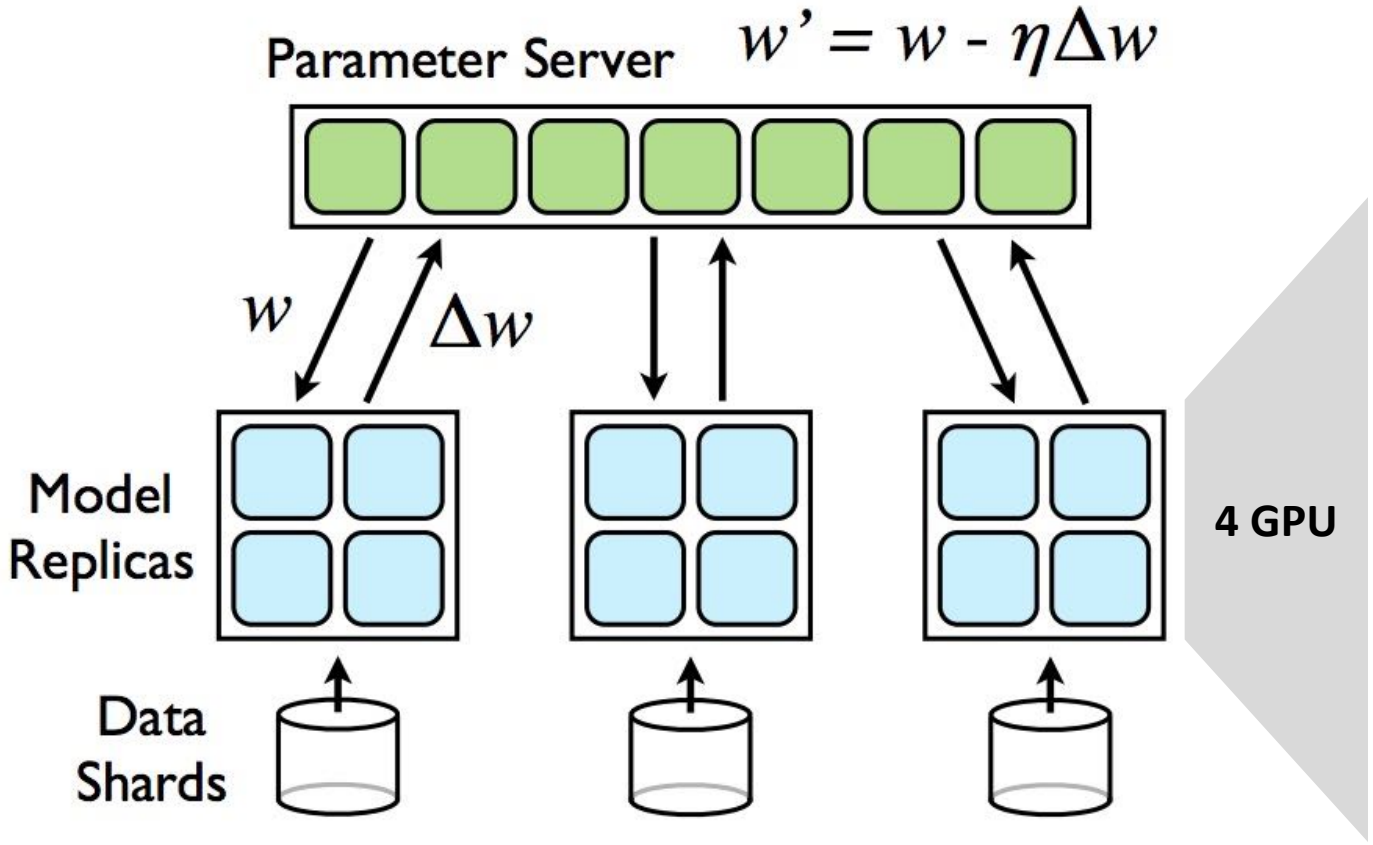
Синхронный SGD: Horovod

- Очень классно скейлится:

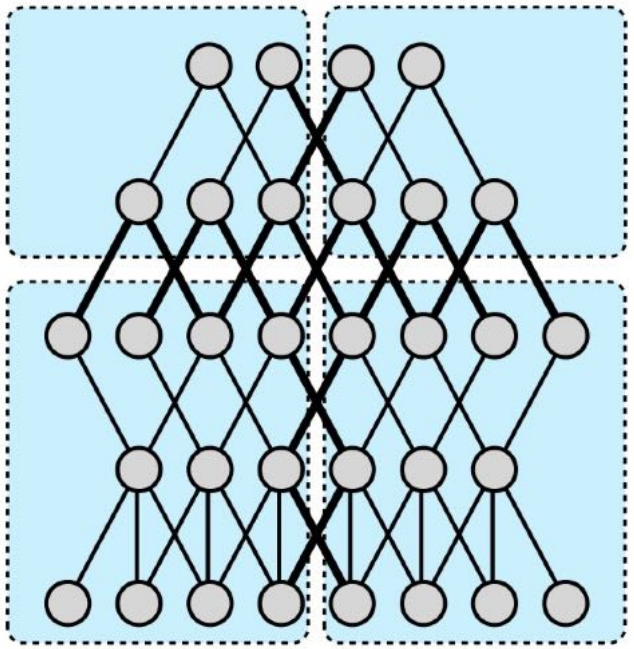


Model-parallel (мало кому нужен)

- Режем веса на партиции (можем их обновлять асинхронно)



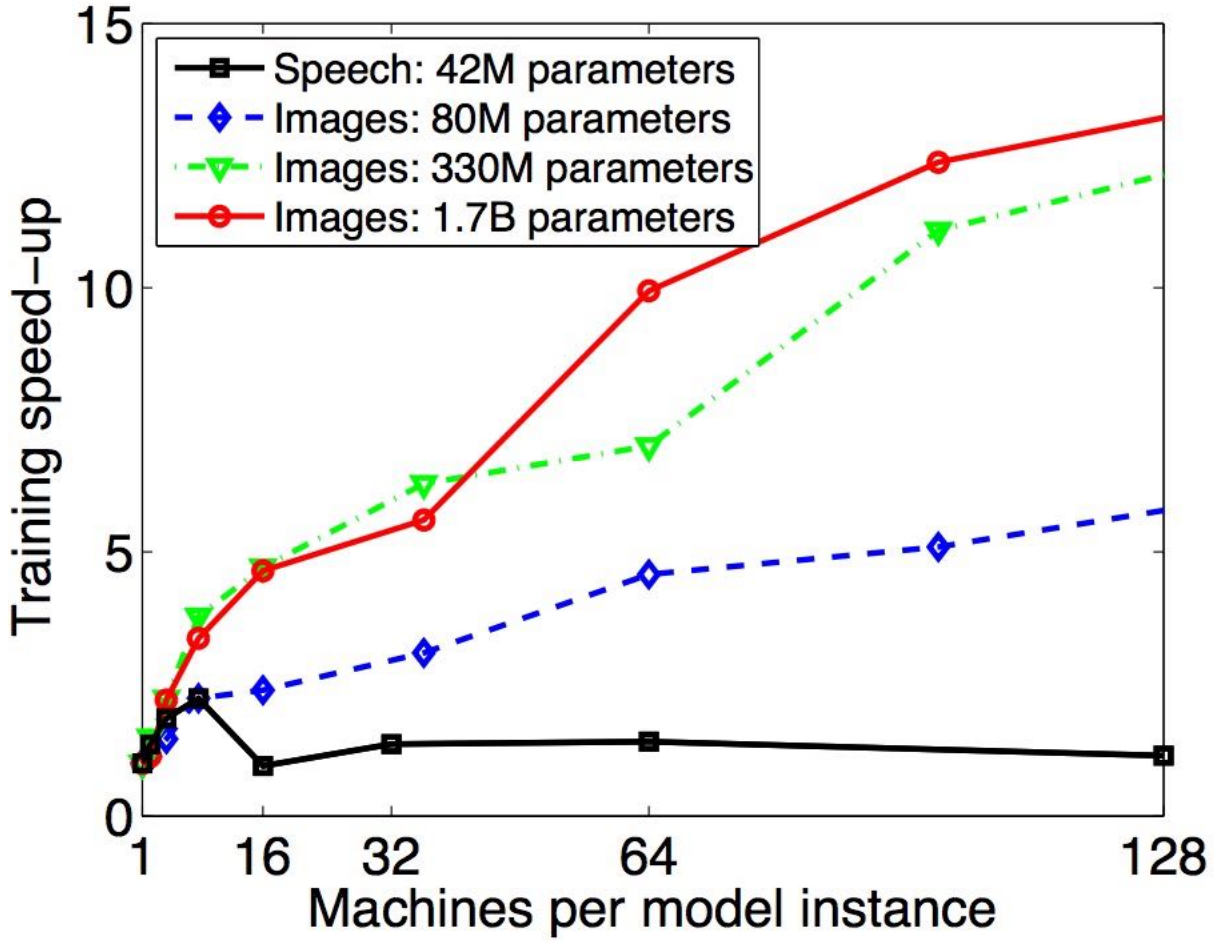
Для сверточных сетей ОК



https://storage.googleapis.com/pub-tools-public-publication-data/pdf/40565.pdf

Model-parallel в DistBelief

Для сверточных сетей ОК



Побили в свое время
state-of-the-art
на большом ImageNet

Model-parallel

- Сложнее реализовывать (нужны выходы нейронов от соседа)
- Не всегда нужно, как правило модель не очень большая

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