Распределенное обучение нейросетей

И.Н.Притуляк

Содержание

- Распределённое обучение
- Оптимизации в распределённом обучении
- CPU / GPU / TPU, CUDA

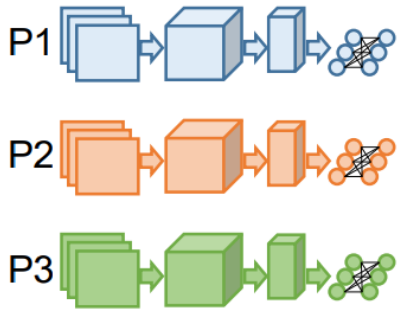
Model и Data Parallelism

Проблема:

- Много данных
- Сложные модели

Решение:

- Model Parallelism
- Data Parallelism



(a) Data Parallelism

(b) Model Parallelism

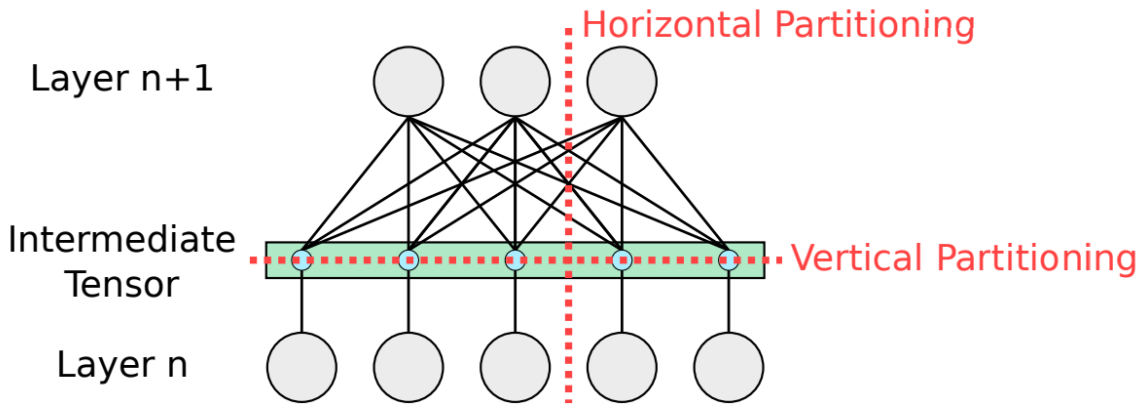
Model Parallelism

Horizontal Partitioning:

- Разные части слоя могут лежать на разных узлах

Vertical Partitioning:

- В наивной реализации много ждём (решение — Pipeline Parallelism)



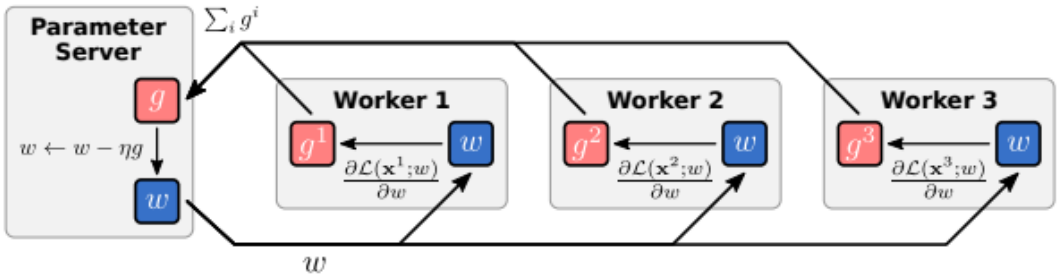
Data Parallelism: Centralized Optimization

Подход:

- Считаем градиенты на разных узлах
- Обновляем параметры на отдельном сервере

Проблемы:

- Все запросы поступают одному узлу
- Ждём, пока все градиенты обработаются



PARAMETER SERVER PROGRAM

Require: initial model $\tilde{w}_0$, learning rate η, number of workers n

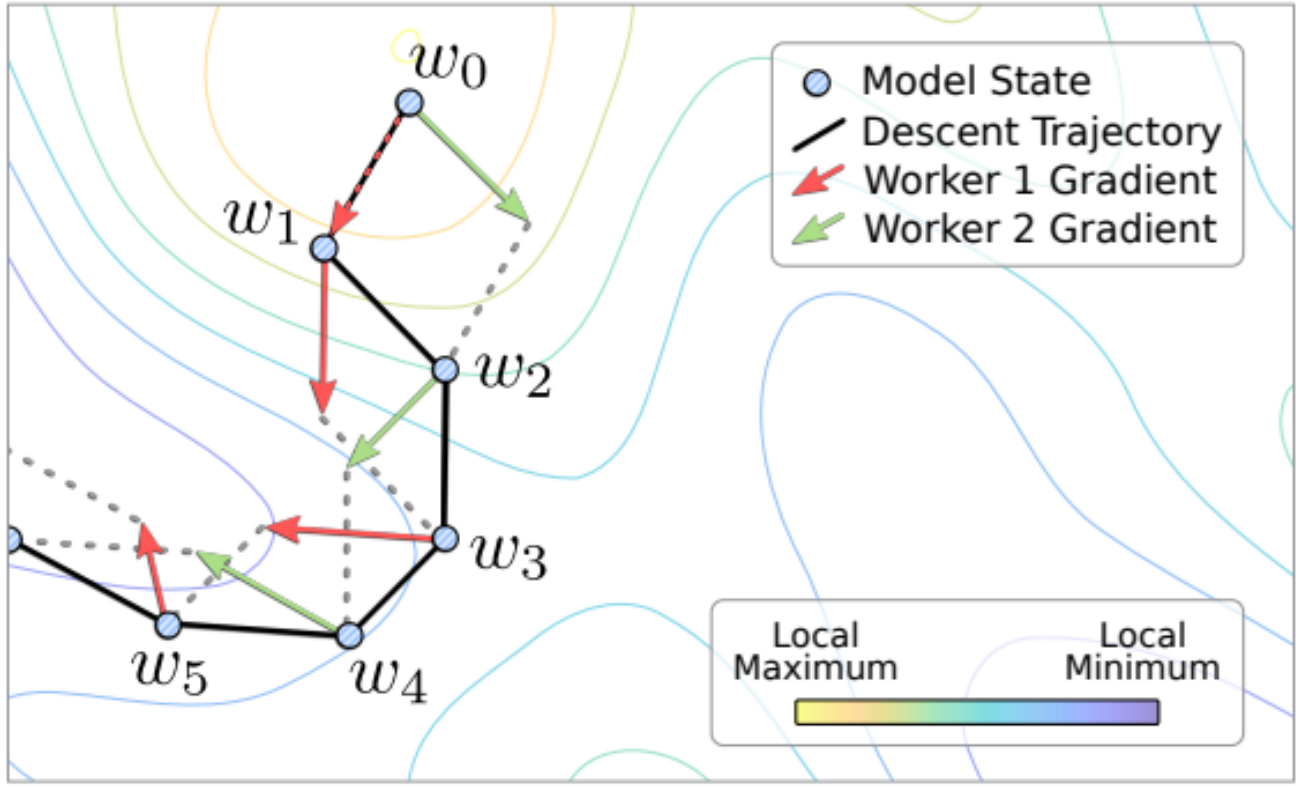
1: **for** $t \leftarrow 0, 1, 2, \ldots$ **do**
2: Broadcast $\tilde{w}_t$
3: Await gradients g_t^i from all workers
4: $\tilde{w}_{t+1} \leftarrow \tilde{w}_t - \eta \sum_{i=1}^{n} g_t^i$
5: **end for**

PROGRAM OF THE i^{th} WORKER

Require: training data source $\mathcal{D}^i$

1: **for** $t \leftarrow 0, 1, 2, \ldots$ **do**
2: Await $\tilde{w}_t$
3: Sample mini-batch $\mathbf{x} \sim \mathcal{D}^i$
4: $g_t^i \leftarrow \frac{\partial \mathcal{L}(\mathbf{x};\tilde{w}_t)}{\partial \tilde{w}_t}$
5: Send g_t^i to parameter server
6: **end for**

Data Parallelism: Centralized Optimization (asynchronous version)



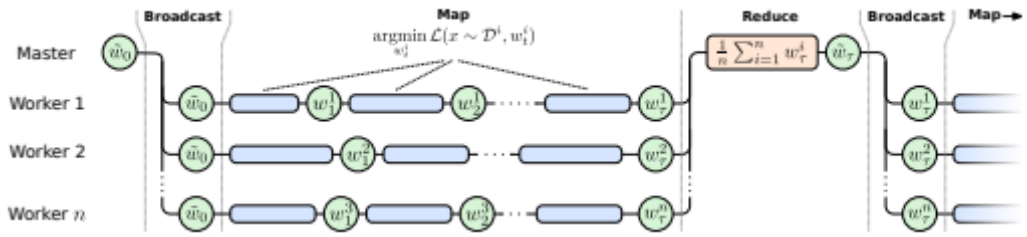
Data Parallelism: Decentralized Optimization

Подход:

- Обучаем модели на разных узлах
- Агрегируем их

Проблемы:

- Подбор количества шагов для обучения
- Ждём, пока все модели обучатся



MASTER PROGRAM

Require: initial model state $\tilde{w}_0$, number of workers n

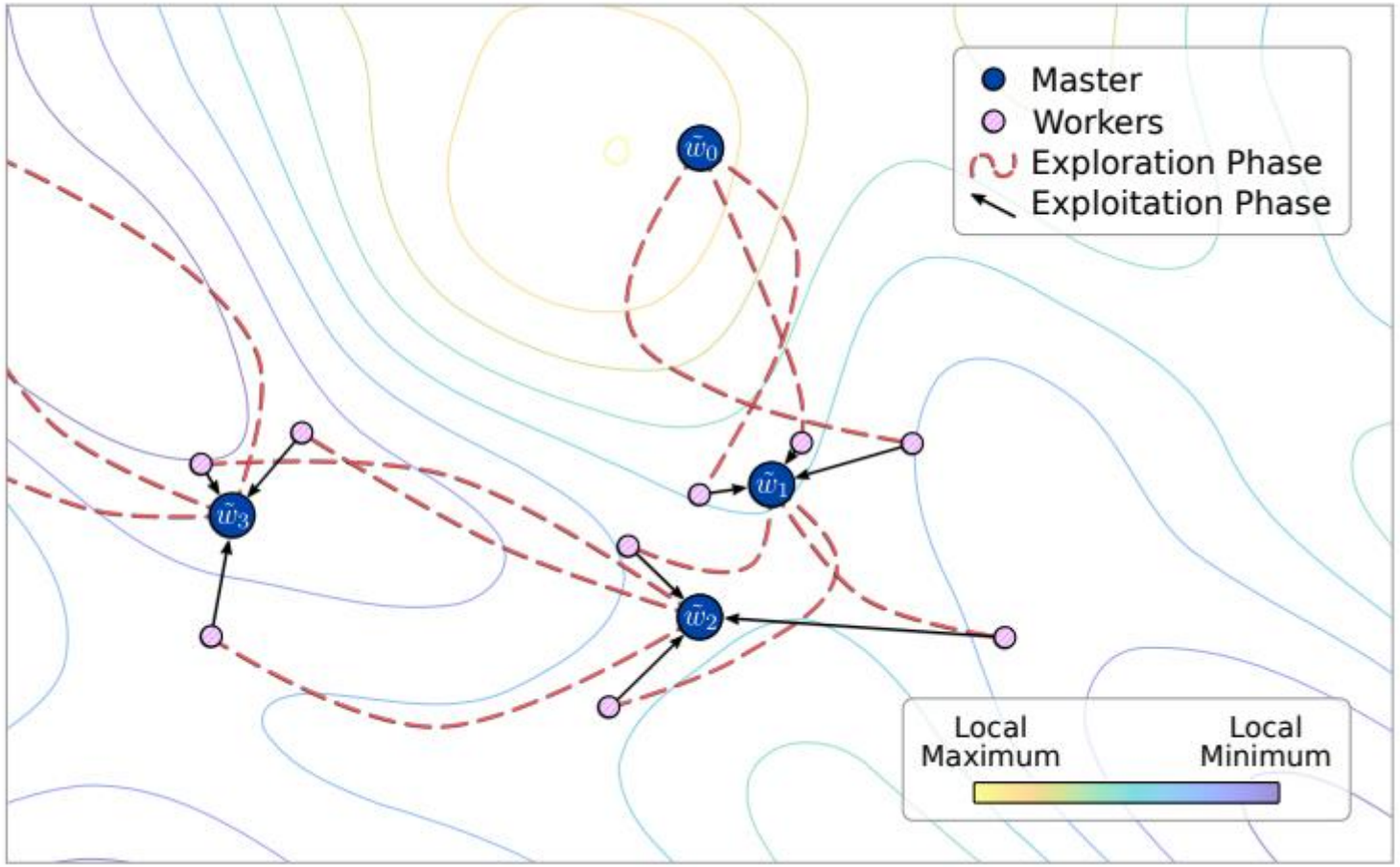
$t \leftarrow 0$
loop
  Broadcast $\tilde{w}_t$
  Await models $w^i_{t+\tau}$ from all workers
  $\tilde{w}_{t+\tau} \leftarrow \frac{1}{n} \sum_{i=1}^{n} w^i_{t+\tau}$
  $t \leftarrow t + \tau$
end loop

PROGRAM RUN BY THE i^{th} WORKER

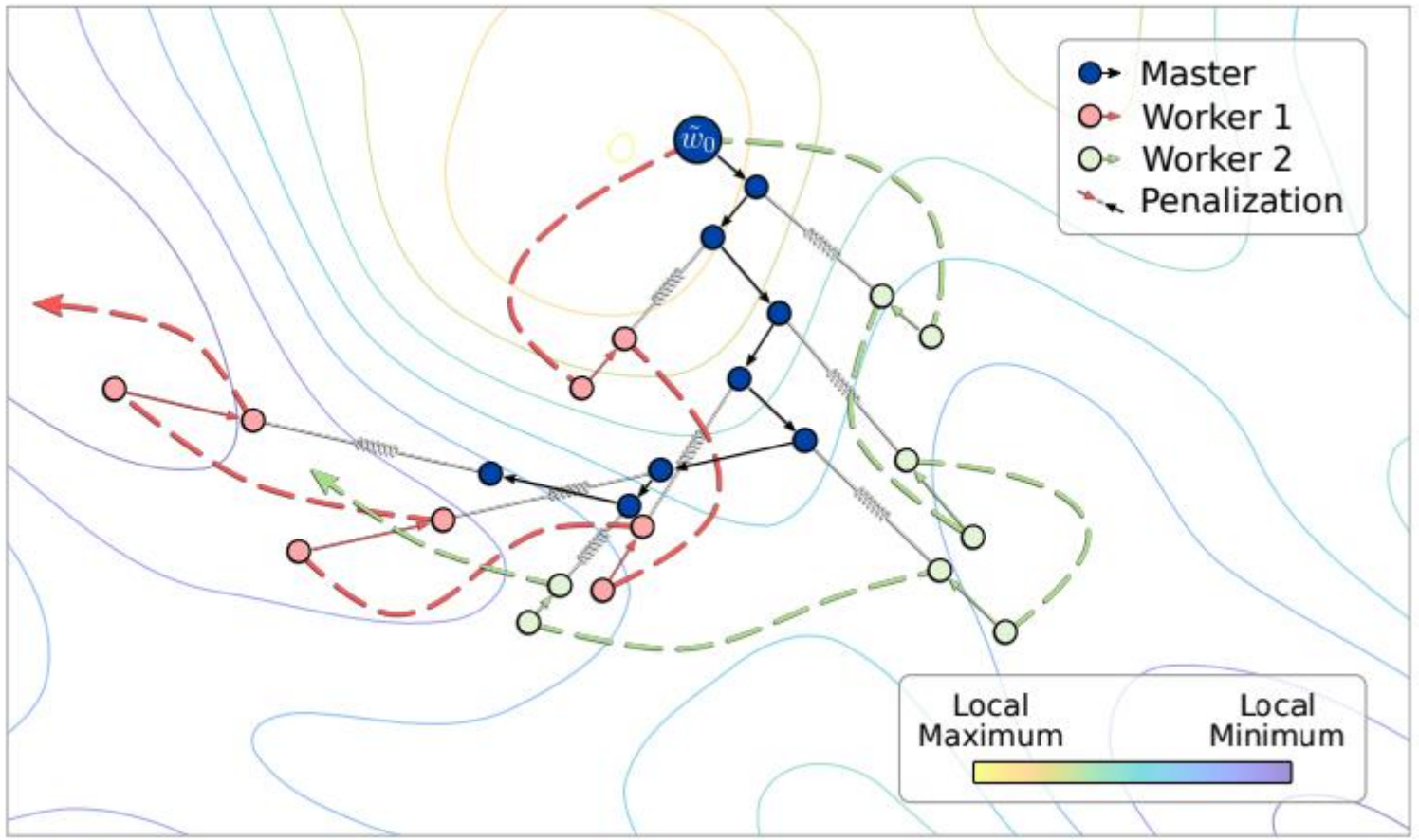
Require: training data source $\mathcal{D}^i$, learning rate η

$t \leftarrow 0$
loop
  Await $\tilde{w}_t$
  $w^i_t \leftarrow \tilde{w}_t$
  for $u \leftarrow t+1, t+2 \ldots, t+\tau$ **do**
    Sample mini-batch $\mathbf{x} \sim \mathcal{D}^i$
    $w^i_{u+1} \leftarrow w^i_u - \eta \frac{\partial \mathcal{L}(\mathbf{x}; w^i_u)}{\partial w^i_u}$
  end for
  Send $w^i_{t+\tau}$ to parameter server
  $t \leftarrow t + \tau$
end loop

Data Parallelism: Decentralized Optimization

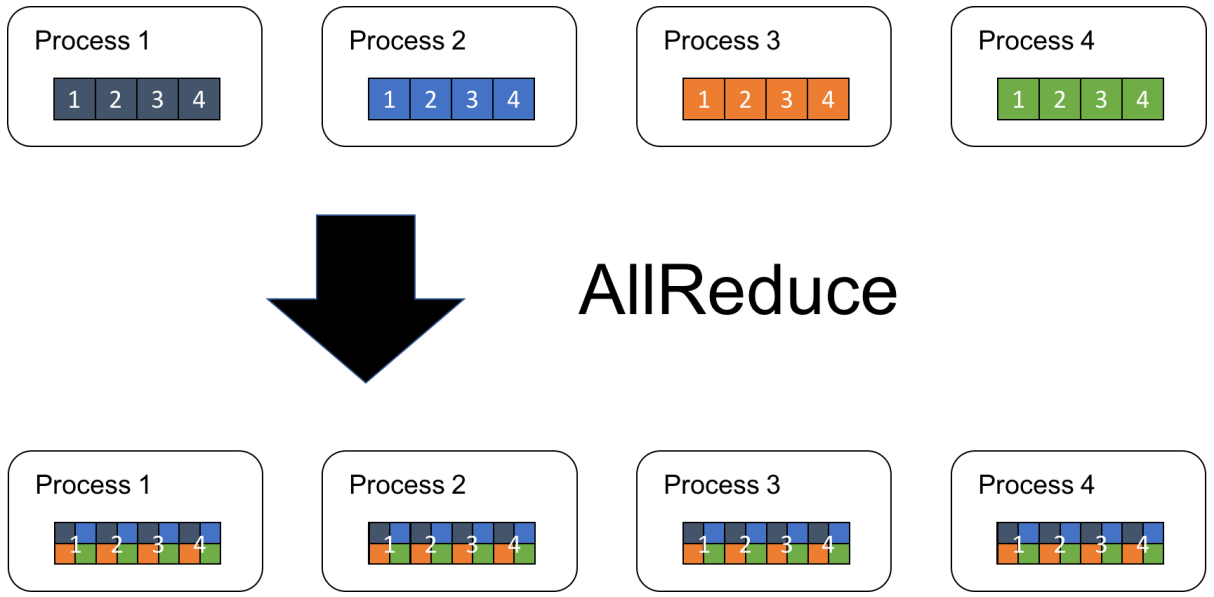


Data Parallelism: Decentralized Optimization (asynchronous version)



AllReduce

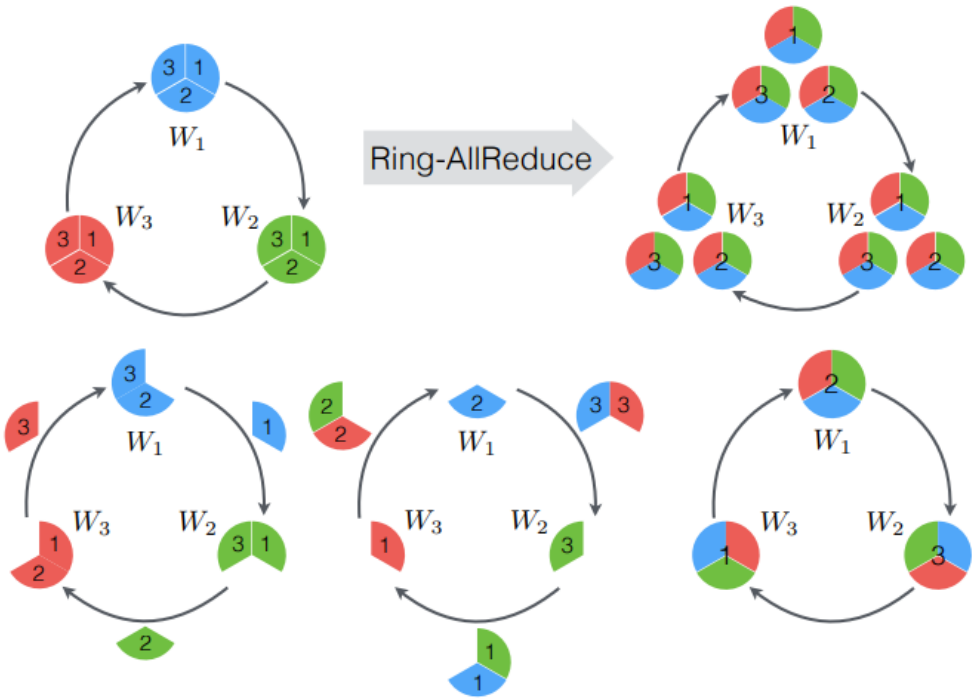
Задача: хотим посчитать сумму градиентов и обновить модель на всех узлах (операция AllReduce).



Ring-AllReduce

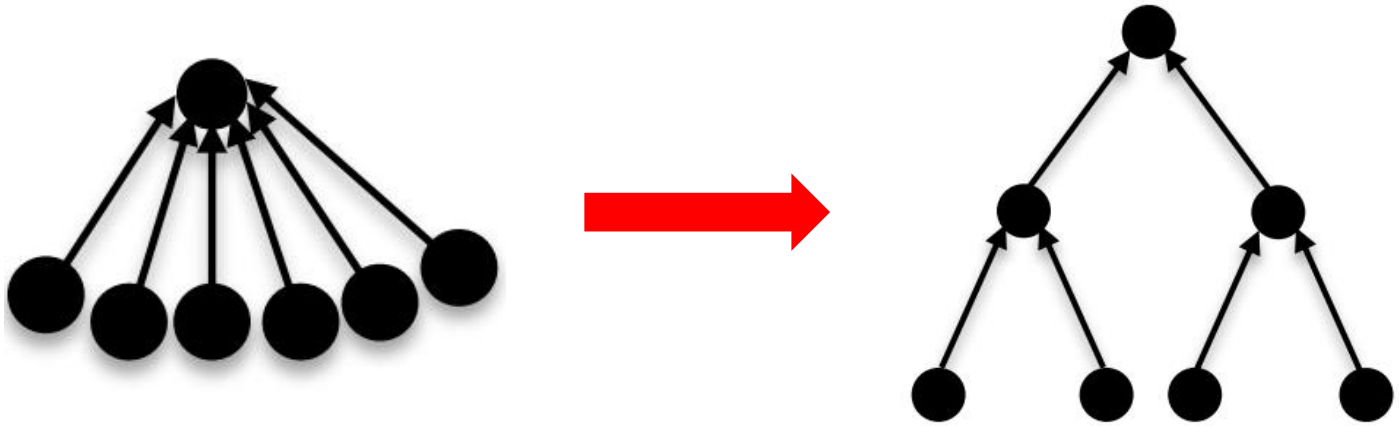
Решение: разбиваем параметры на N групп, аккумулируем соответствующие им градиенты путём передачи их по кругу.

Нагрузка на один узел: $2M(N-1)/N$



Tree-AllReduce

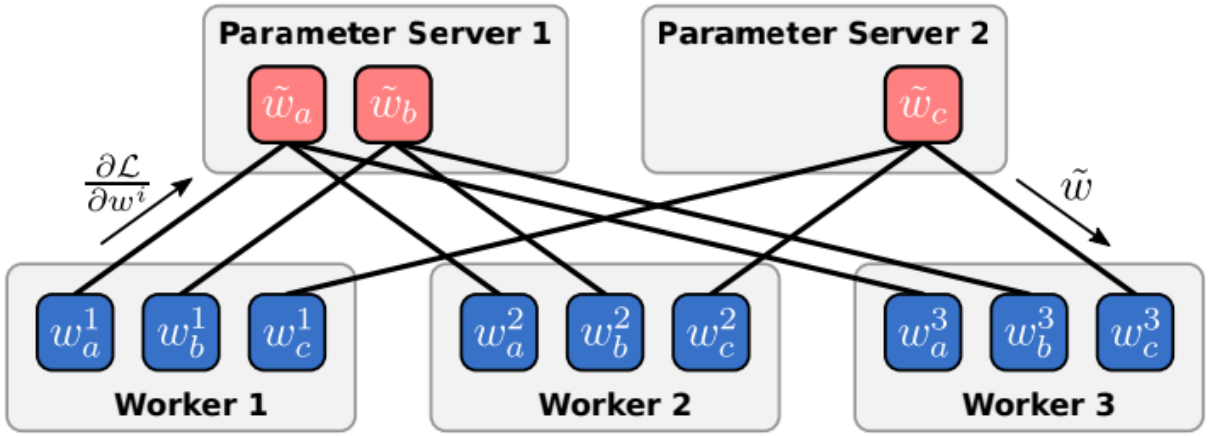
Решение: строим бинарное дерево из узлов и делимся градиентами с предками, итоговое значение затем спускаем вниз.



All-to-all Reduce

Решение:

- Разделим параметры на несколько групп
- Каждой группе выделим отдельный узел
- Будем обрабатывать градиенты на соответствующих узлах



Источники

- https://arxiv.org/ftp/arxiv/papers/2007/2007.03970.pdf
- https://dlsys.cs.washington.edu/pdf/lecture11.pdf
- https://tech.preferred.jp/en/blog/technologies-behind-distributed-deep-learning-allreduce
- https://lilianweng.github.io/lil-log/2021/09/24/train-large-neural-networks.html