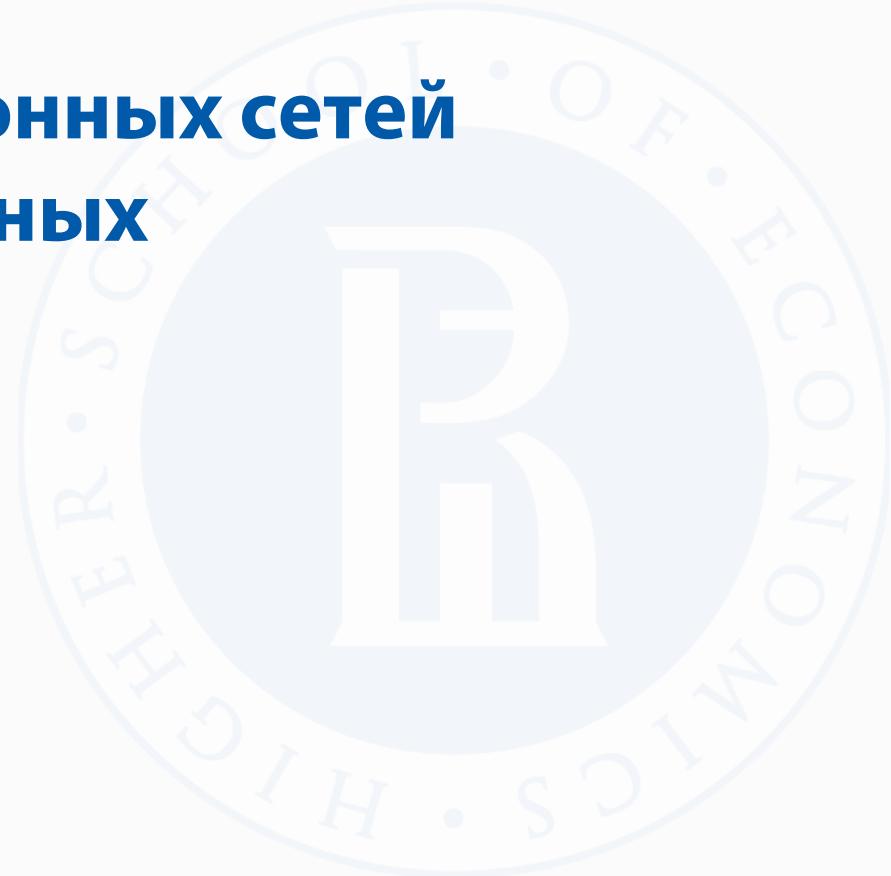


# **Обучение нейронных сетей на больших данных**



# План



Проблемы обучения сетей на больших данных



# План



Проблемы обучения сетей на больших данных



Подходы к решению



# План



Проблемы обучения сетей на больших данных



Подходы к решению



Адаптация процесса обучения:



→ data-parallel



→ model-parallel



# План



Проблемы обучения сетей на больших данных



Подходы к решению



Адаптация процесса обучения:



→ data-parallel



→ model-parallel



Transfer learning



# План



Проблемы обучения сетей на больших данных



Подходы к решению



Адаптация процесса обучения:



→ data-parallel



→ model-parallel



Transfer learning



Другие подходы



# Проблемы больших моделей

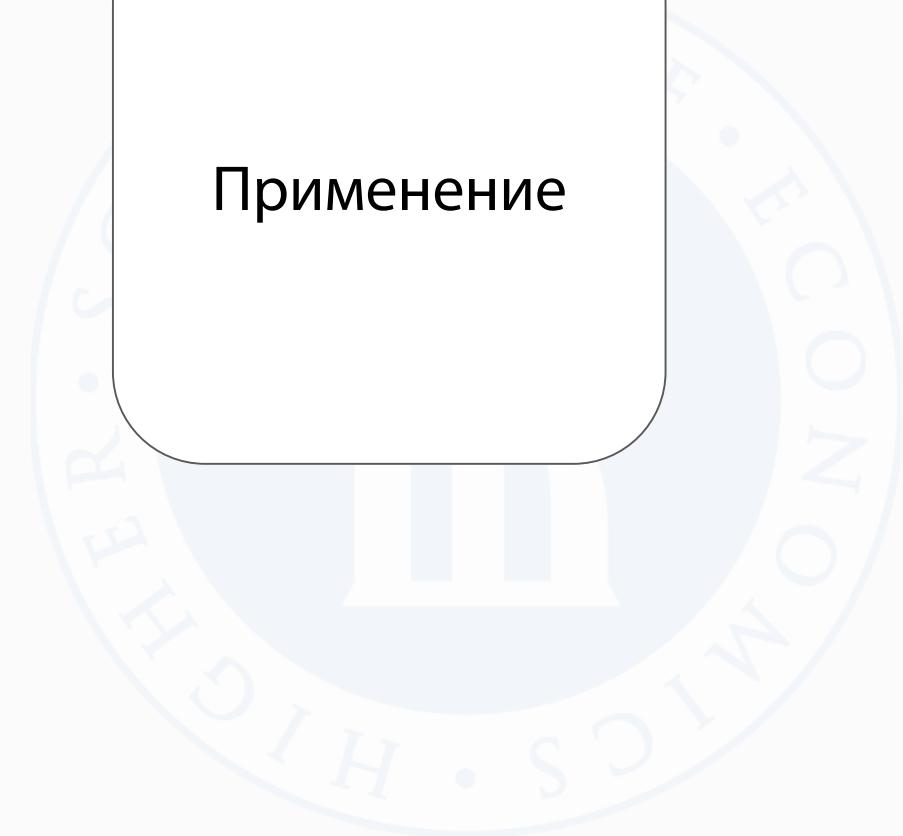
Обучение



# Проблемы больших моделей

Обучение

Применение



# Проблемы больших моделей

Обучение

Применение

# Проблемы обучения на больших данных

→ Две основные проблемы:

# Проблемы обучения на больших данных

- Две основные проблемы:
  - время обучения



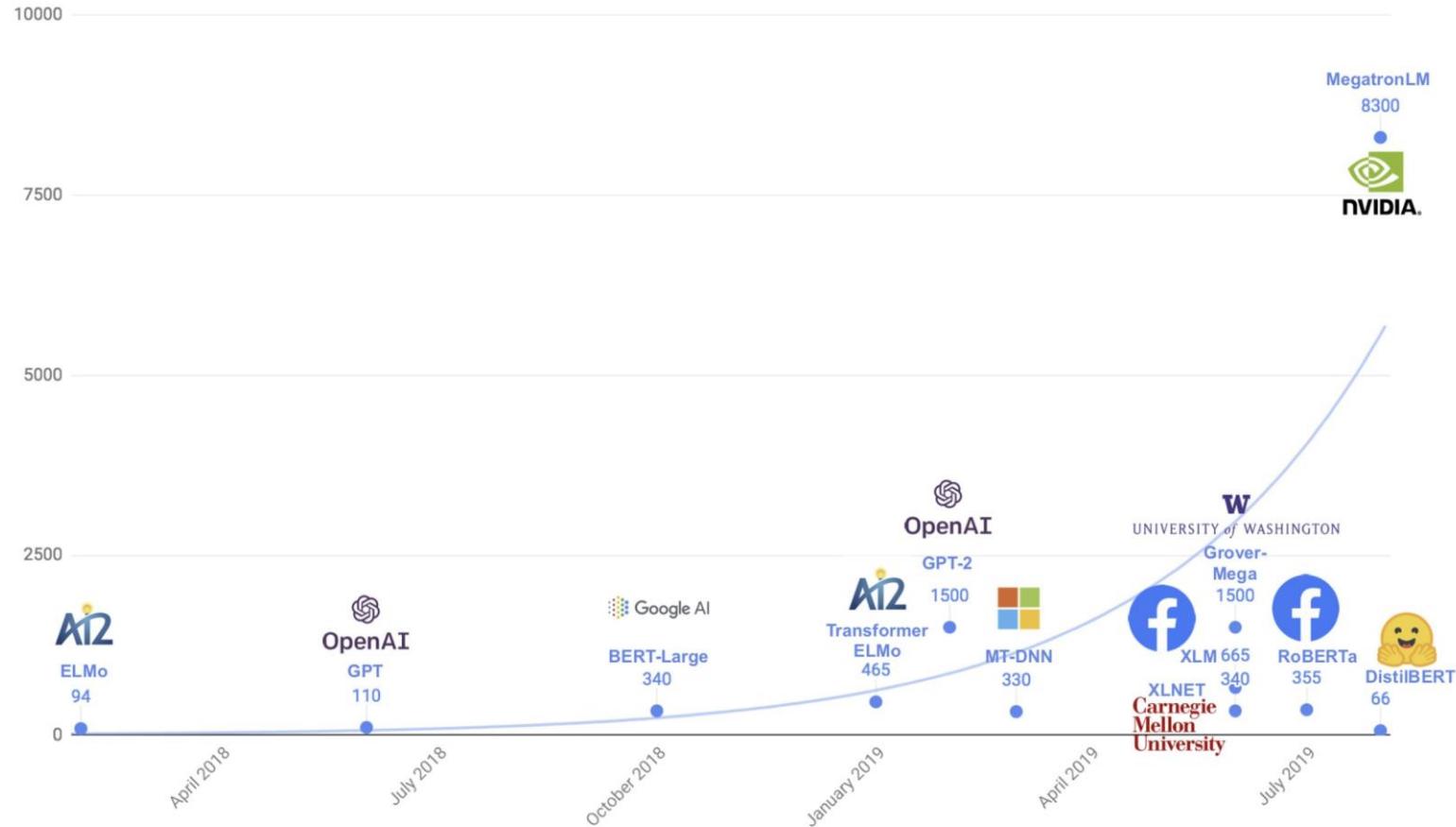
# Проблемы обучения на больших данных

→ Две основные проблемы:

- время обучения
- размер модели и данных



# Тенденция



# Пример. OpenAI GPT

→ GPT — Generative Pre-trained Transformer



# Пример. OpenAI GPT

- GPT — Generative Pre-trained Transformer
- Алгоритм обработки естественного языка



# Пример. OpenAI GPT

- GPT — Generative Pre-trained Transformer
- Алгоритм обработки естественного языка
- На сентябрь 2020 — самая крупная и продвинутая языковая модель в мире



# Пример. OpenAI GPT

- GPT — Generative Pre-trained Transformer
- Алгоритм обработки естественного языка
- На сентябрь 2020 — самая крупная и продвинутая языковая модель в мире
- По заявлению разработчиков — решает «любые задачи на английском языке»



# Пример. GPT-3

→ Перевод текста в команды Linux

```
~ # cmdxyz turns text into Linux commands.  
~ # Built with OpenAI using GPT-3.  
  
~ # cmdxyz create a directory named foo, and enter it  
~ # mkdir foo; cd foo;  
~ # cmdxyz create a file named test.txt that contains 3 colors  
~ # echo "red green blue" > test.txt  
~ # cmdxyz list files in this directory  
~ # ls  
~ # test.txt
```



# Пример. GPT-3

## → Аналогии

Q: If a b c changes to a b d , what does p q r change to?

A: p q s

Q: If a b c changes to a b d , what does i j k l m change to?

A: i j k l n

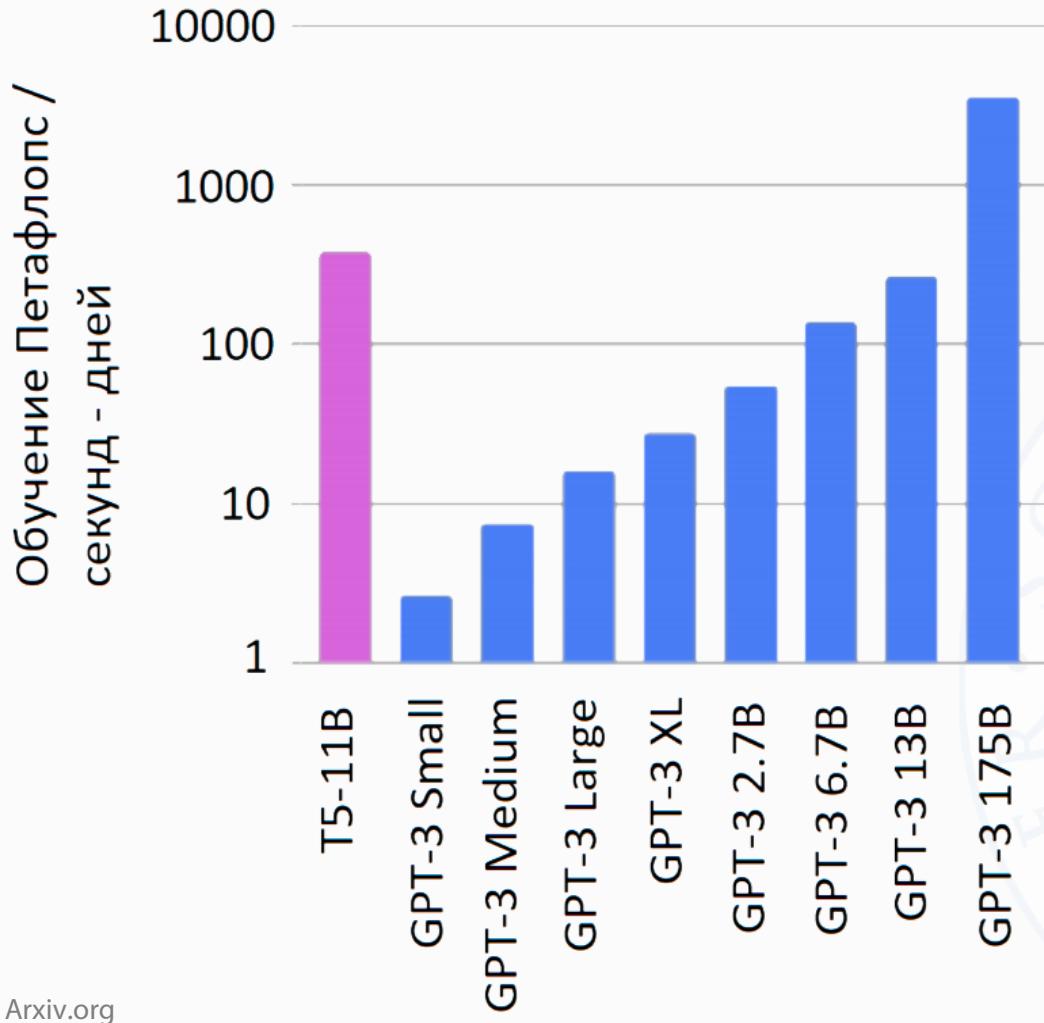
Q: If a b c changes to a b d , what does r s t u v w change to?

A: r s t u v x

Q: If a b c changes to a b d , what does e f g h i j k change to?

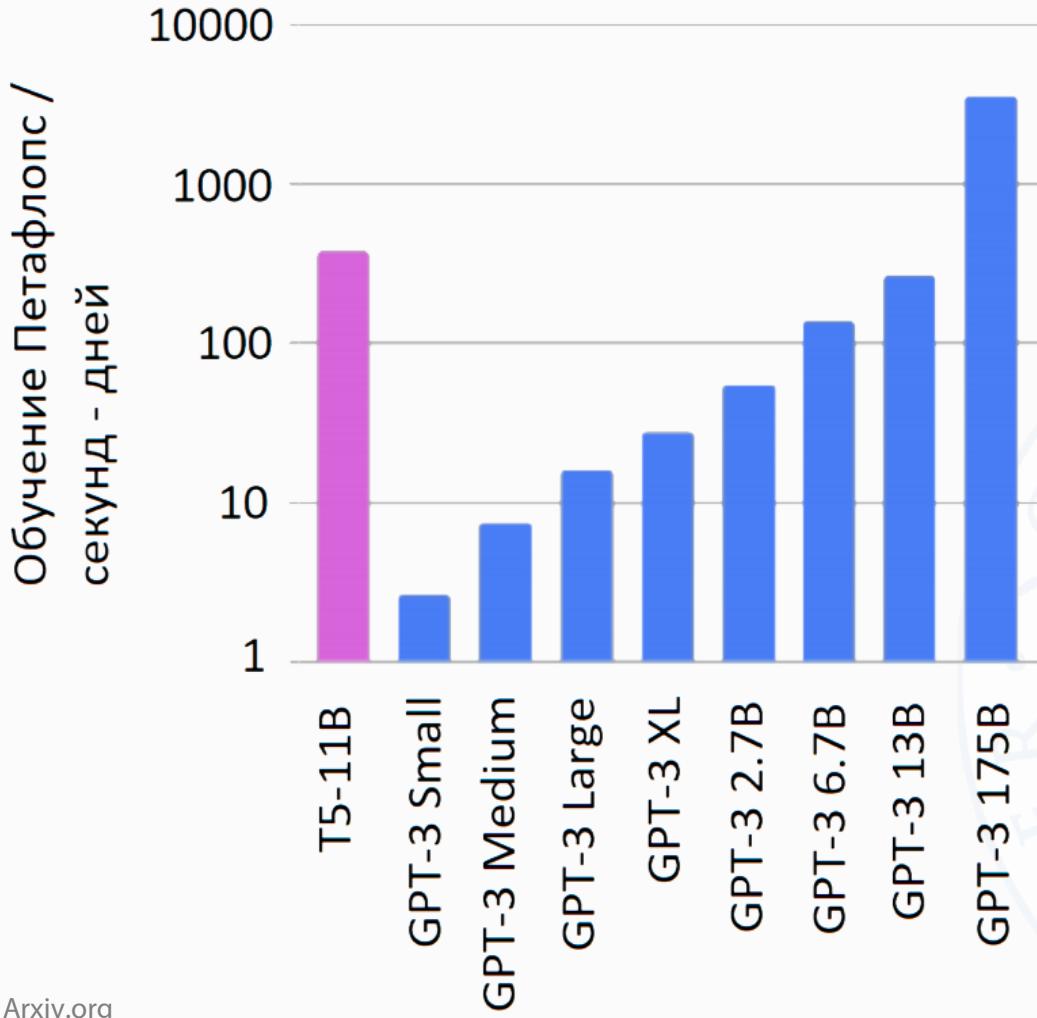
# Пример. GPT-3

→ 175 000 000 000 параметров



# Пример. GPT-3

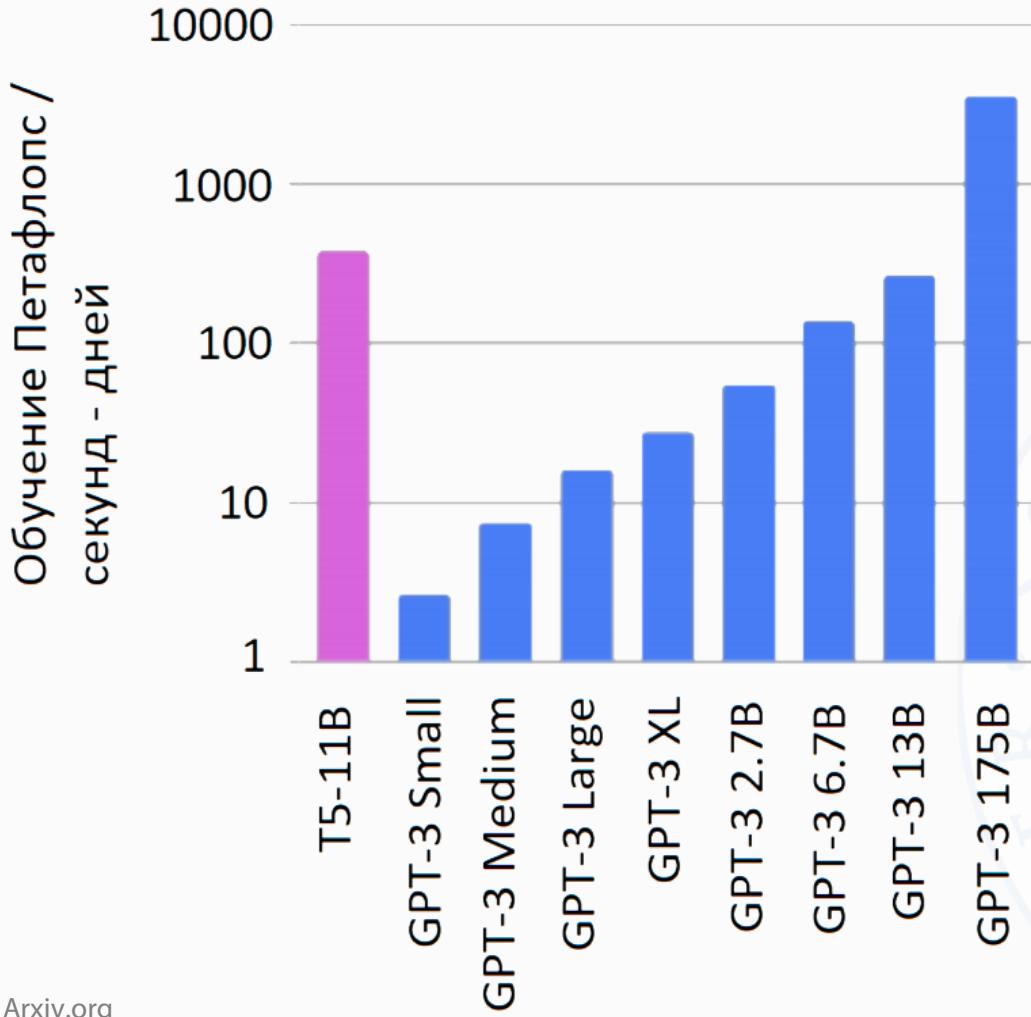
→ 175 000 000 000 параметров



→ ~700GB данных при использовании FP32

# Пример. GPT-3

→ 175 000 000 000 параметров

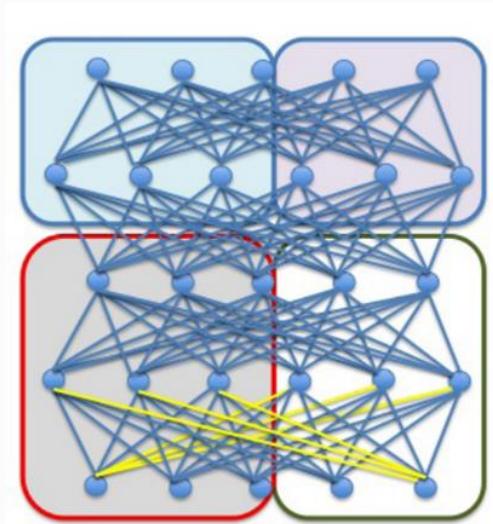


→ ~700GB данных при использовании FP32

→ 355 лет обучения на одной GPU Tesla V100

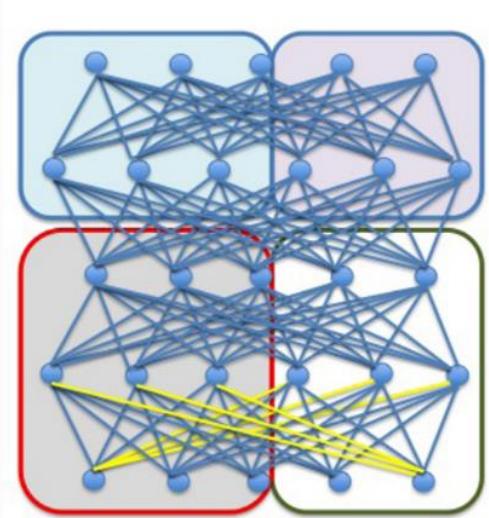
# Подходы к решению

→ Адаптация обучения — масштабирование



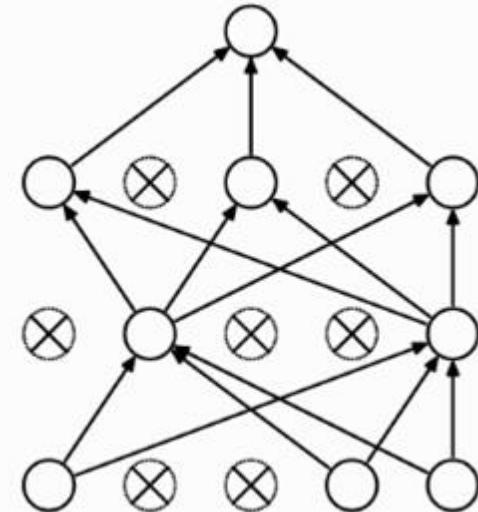
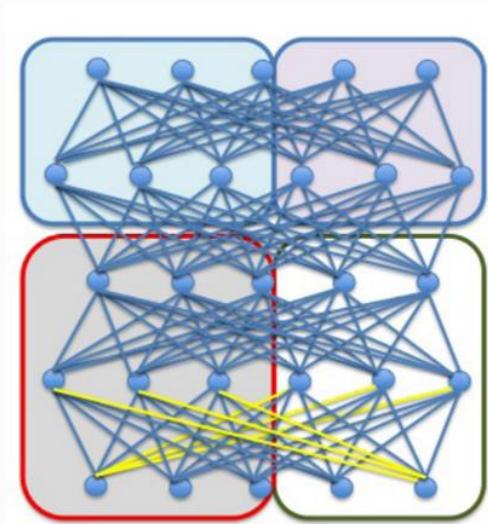
# Подходы к решению

- Адаптация обучения — масштабирование
- Передача обучения (Transfer learning)



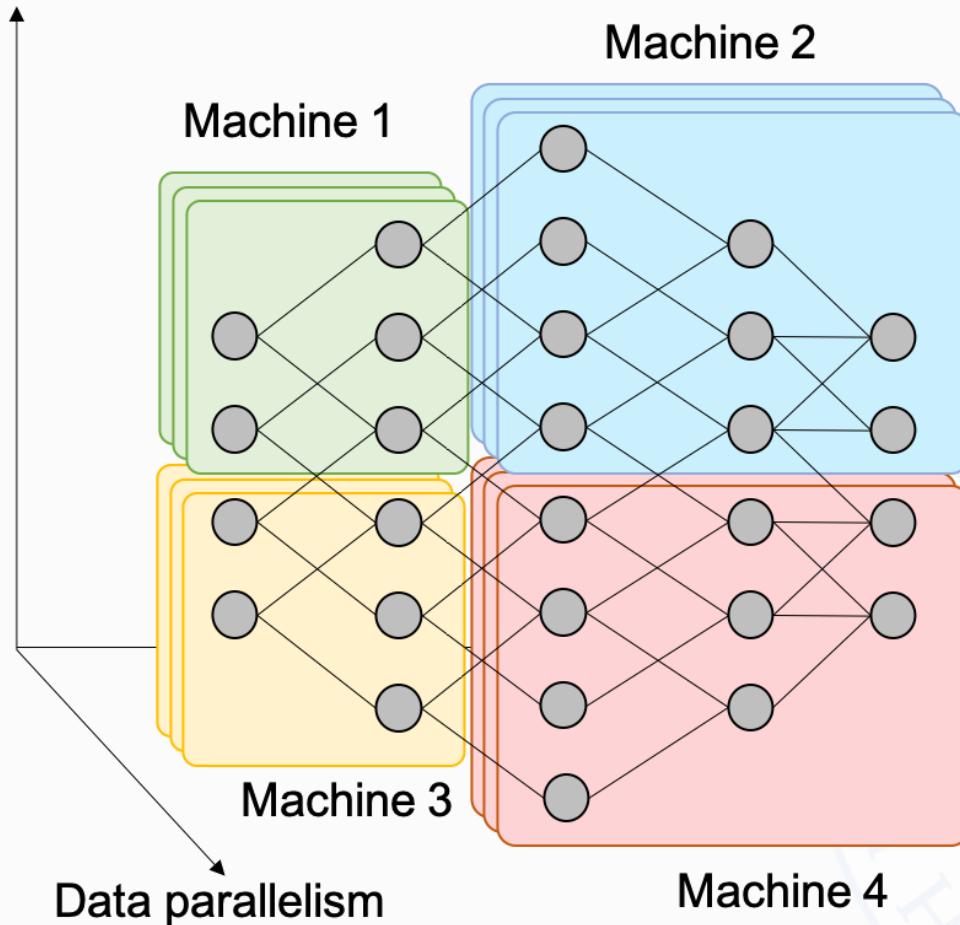
# Подходы к решению

- Адаптация обучения — масштабирование
- Передача обучения (Transfer learning)
- Другие подходы — изменение модели — прореживание, квантизация



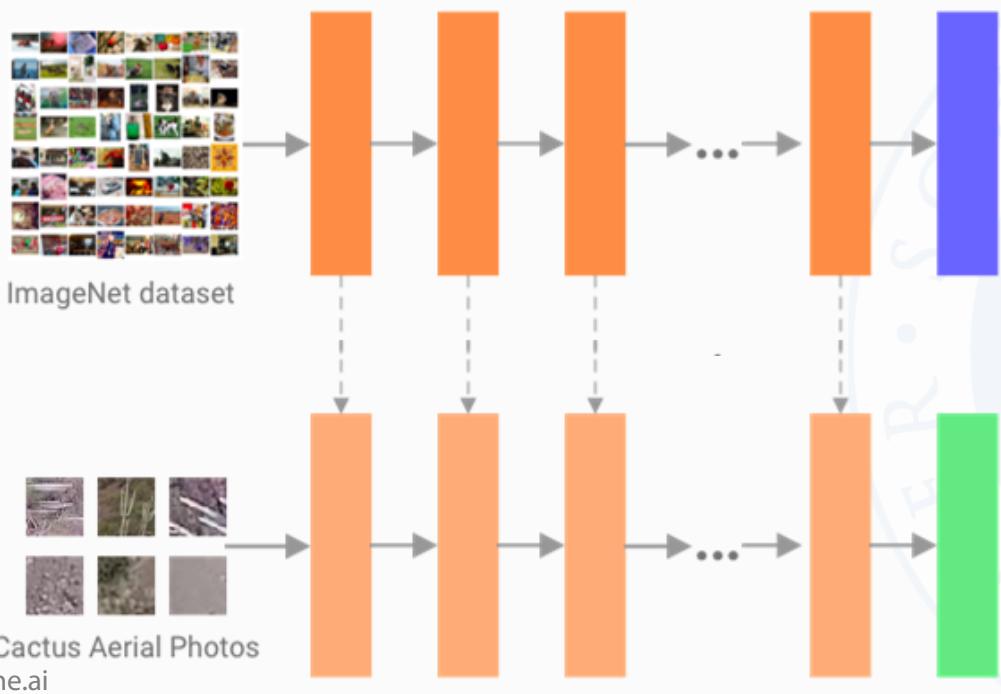
# Адаптация обучения

Model parallelism



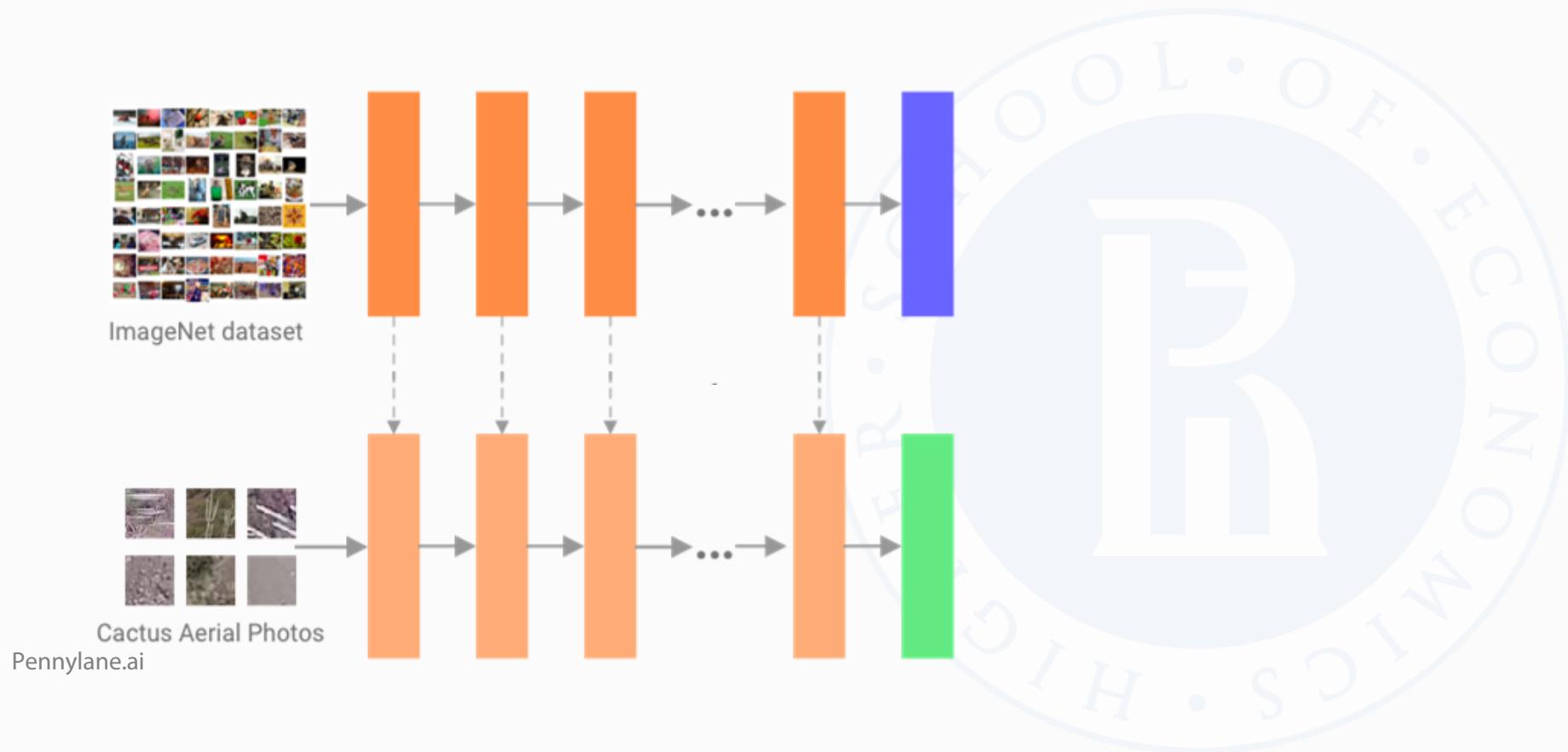
# Передача обучения

→ Использование другой модели для решения своих задач:



# Передача обучения

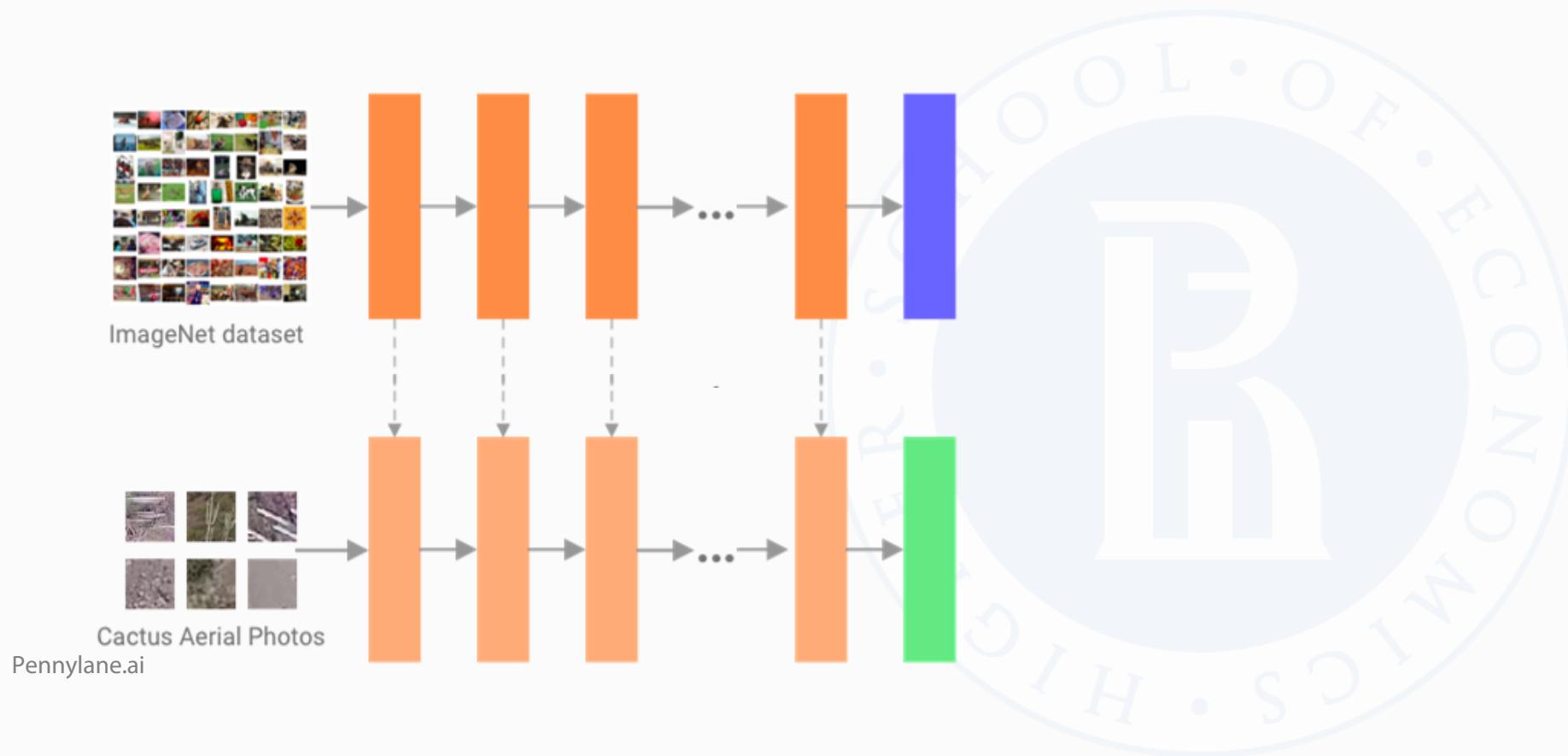
- Использование другой модели для решения своих задач:
- Дообучение (fine-tuning) всей модели



# Передача обучения

→ Использование другой модели для решения своих задач:

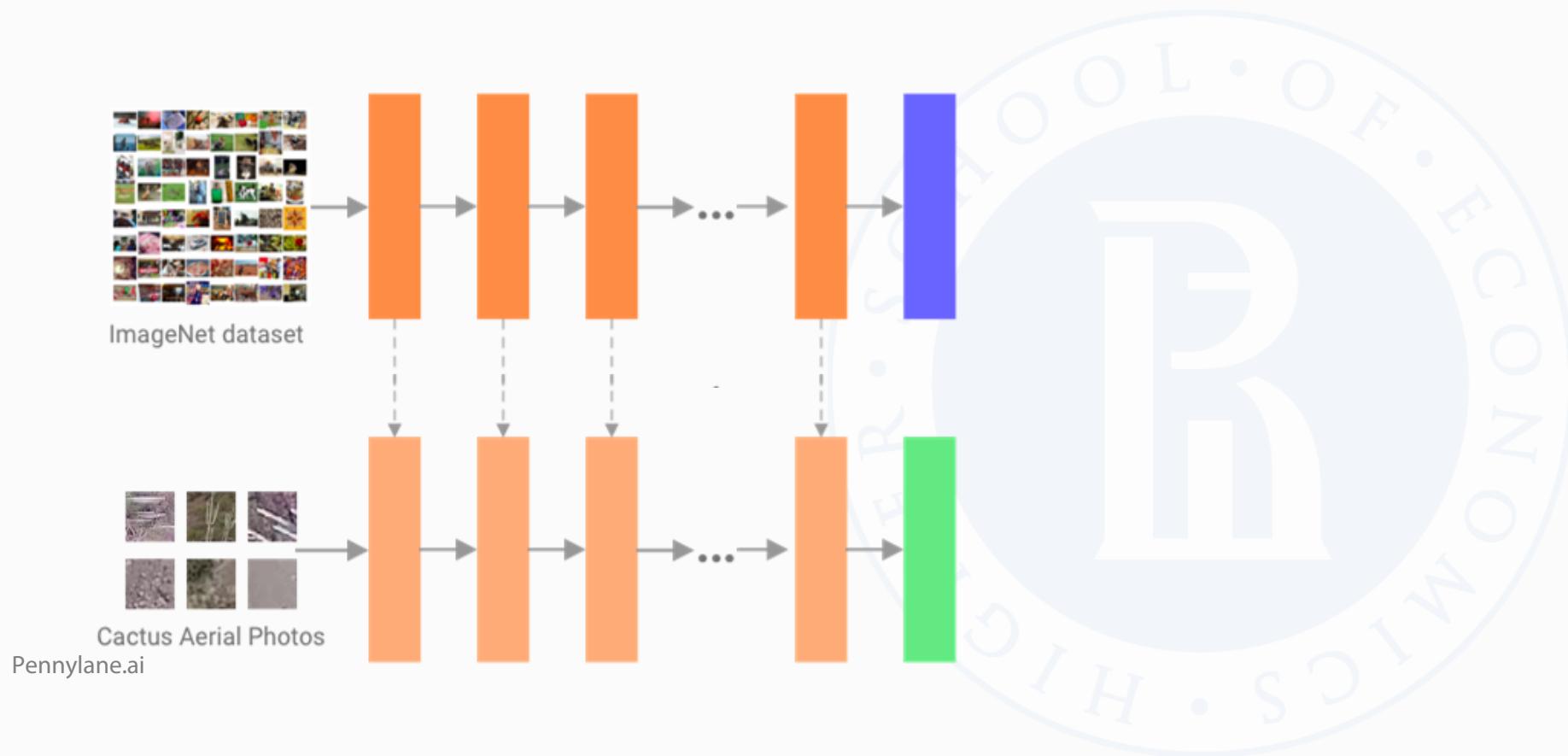
- Дообучение (fine-tuning) всей модели
- Дообучение выходных слоев



# Передача обучения

→ Использование другой модели для решения своих задач:

- Дообучение (fine-tuning) всей модели
- Дообучение выходных слоев
- Извлечение признаков

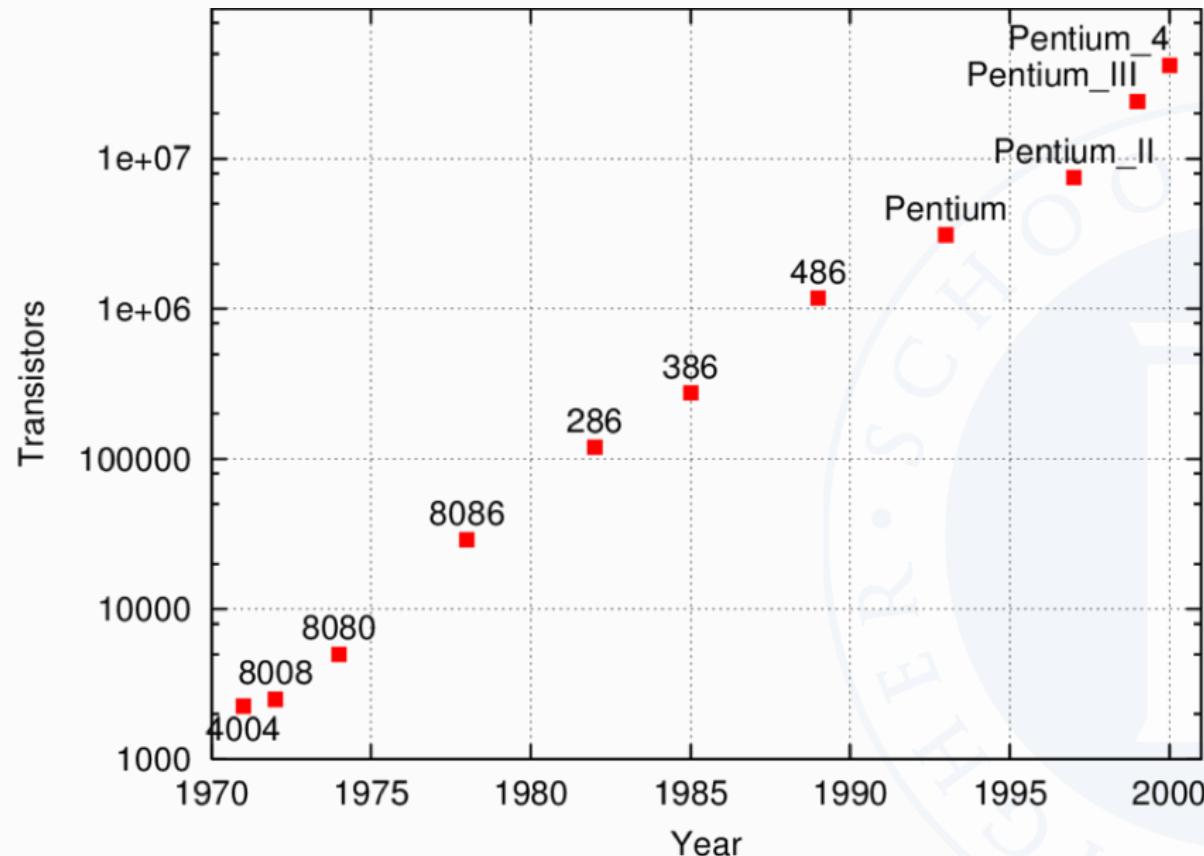


# Другие подходы



Очевидные варианты:

- увеличение мощности CPU, GPU, ТРУ

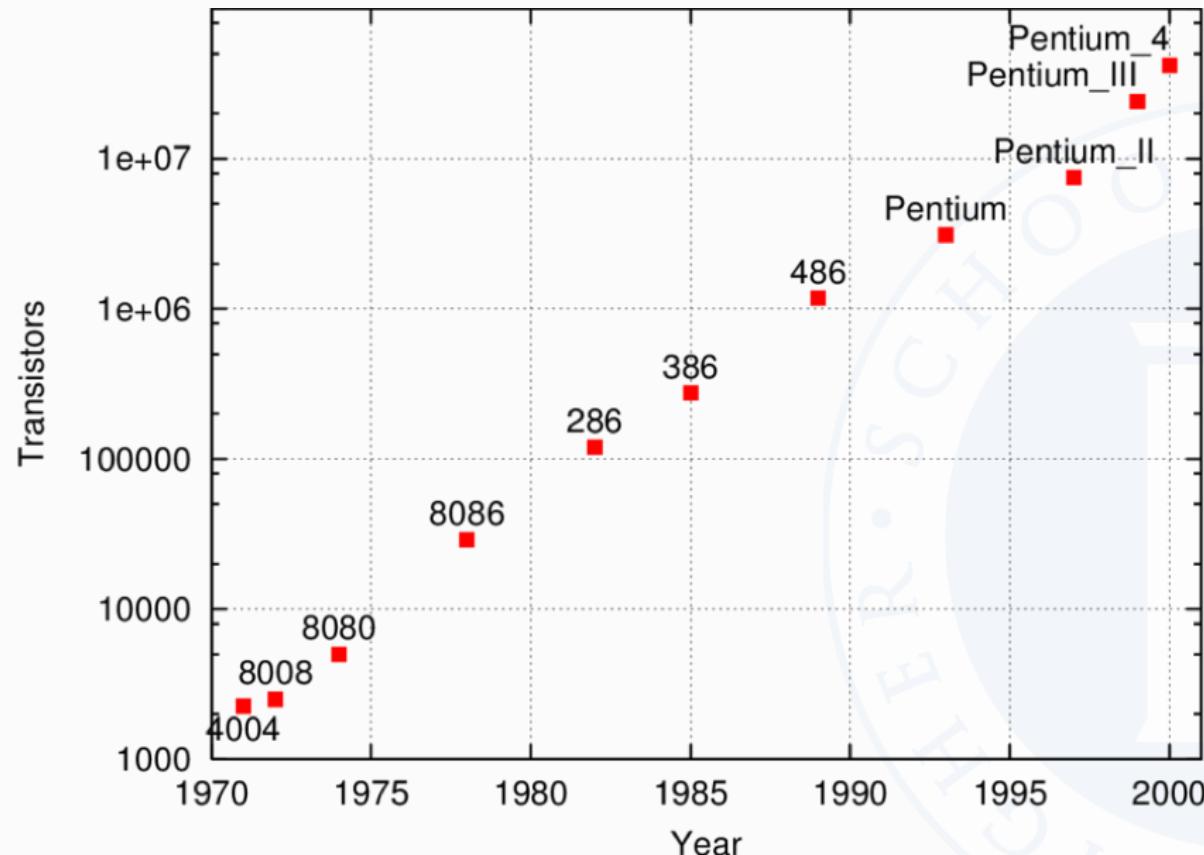


# Другие подходы



Очевидные варианты:

- увеличение мощности CPU, GPU, ТРУ
- использование более простых моделей



# Другие подходы



Очевидные варианты:

- увеличение мощности CPU, GPU, TPU
- использование более простых моделей

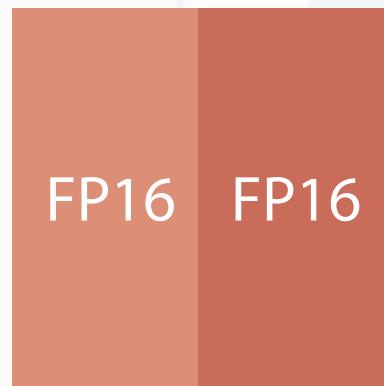


Изменение используемых типов данных:

- частичная квантизация, прореживание



OR



(Same Op)

# Другие подходы



Очевидные варианты:

- увеличение мощности CPU, GPU, TPU
- использование более простых моделей



Изменение используемых типов данных:

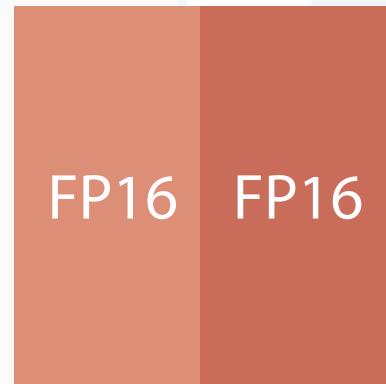
- частичная квантизация, прореживание



В курсе «Проектирование и реализация систем машинного обучения»



OR

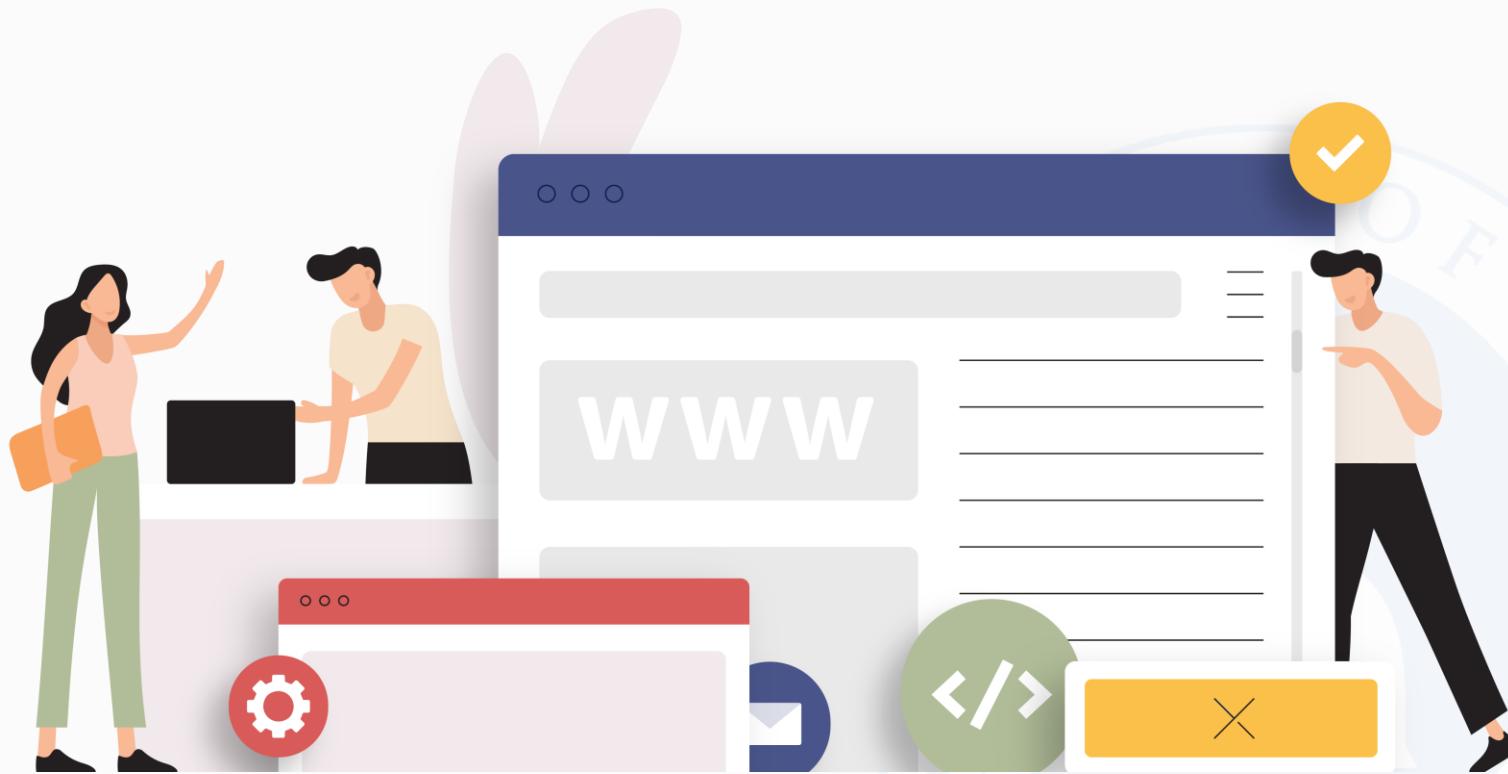


(Same Op)

# Далее



В следующем видео поговорим о подходах к изменению процесса обучения на одном компьютере при наличии нескольких устройств для обработки данных



# **Масштабирование обучения нейронных сетей**



# План



Процесс обучения



# План



Процесс обучения



Параллельность по модели



# План



Процесс обучения



Параллельность по модели



Параллельность по данным



# План



Процесс обучения



Параллельность по модели



Параллельность по данным



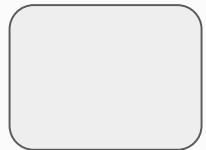
Терминология MPI



# Процесс обучения

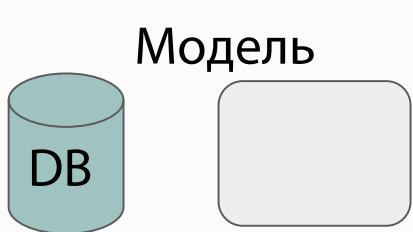
1. Формирование некоторой архитектуры сети, модели

Модель



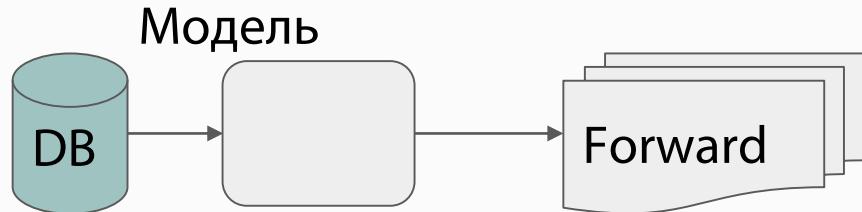
# Процесс обучения

1. Формирование некоторой архитектуры сети, модели
2. Использование определенного набора данных



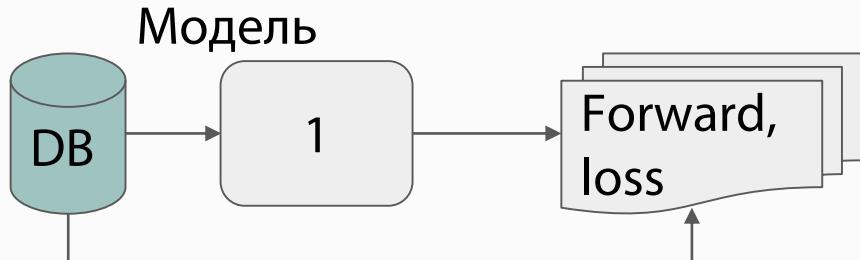
# Процесс обучения

1. Формирование некоторой архитектуры сети, модели
2. Использование определенного набора данных
3. Части данных (batch) проходят через модель (forward)



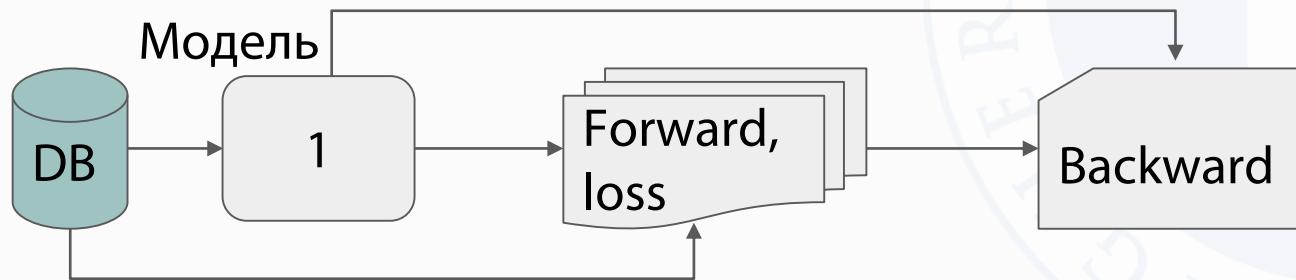
# Процесс обучения

1. Формирование некоторой архитектуры сети, модели
2. Использование определенного набора данных
3. Части данных (batch) проходят через модель (forward)
4. Для полученных результатов считается ошибка



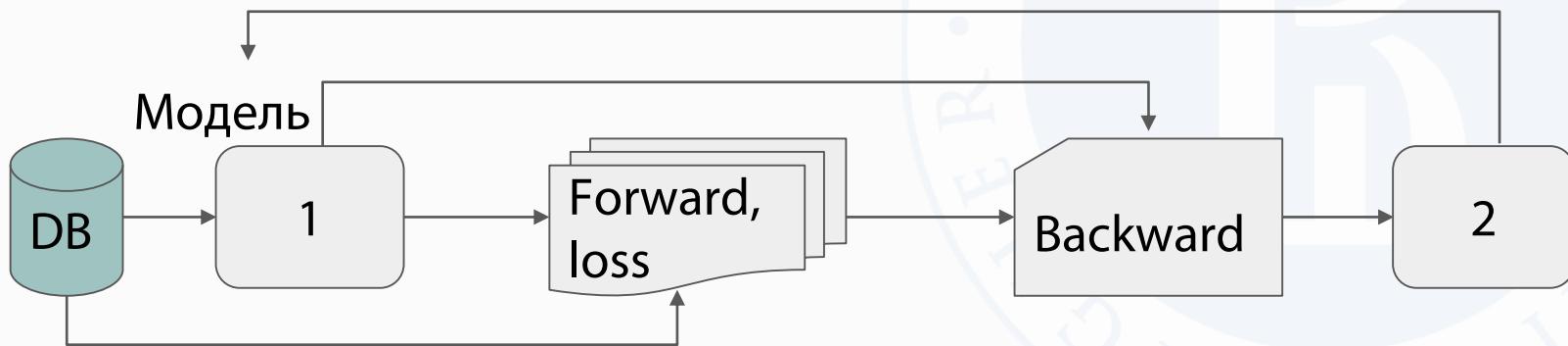
# Процесс обучения

1. Формирование некоторой архитектуры сети, модели
2. Использование определенного набора данных
3. Части данных (batch) проходят через модель (forward)
4. Для полученных результатов считается ошибка
5. Считается градиент (backward, back propagation)



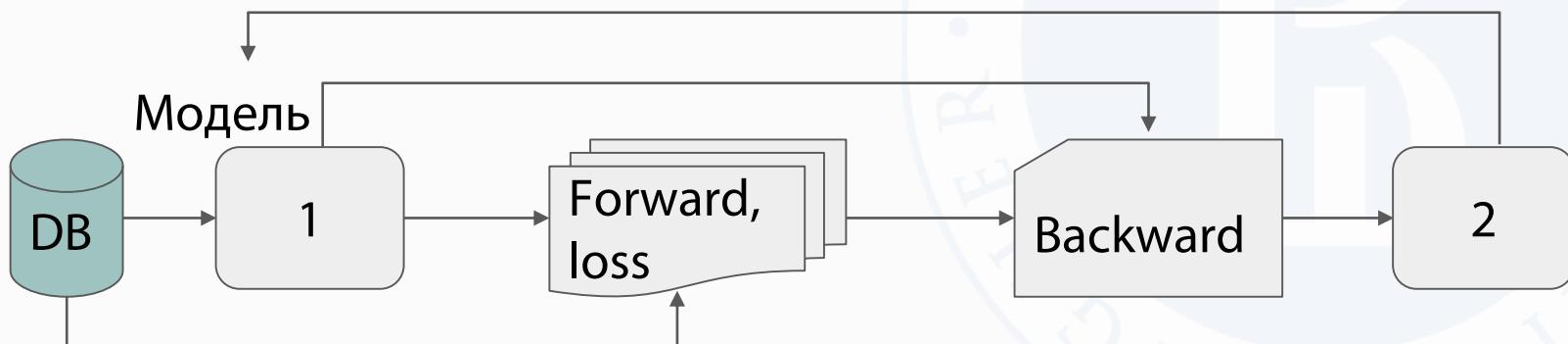
# Процесс обучения

1. Формирование некоторой архитектуры сети, модели
2. Использование определенного набора данных
3. Части данных (batch) проходят через модель (forward)
4. Для полученных результатов считается ошибка
5. Считается градиент (backward, back propagation)
6. Изменяются веса, обновляется модель



# Процесс обучения

1. Формирование некоторой архитектуры сети, модели
2. Использование определенного набора данных
3. Части данных (batch) проходят через модель (forward)
4. Для полученных результатов считается ошибка
5. Считается градиент (backward, back propagation)
6. Изменяются веса, обновляется модель
7. Повторяются п. 3-6 до достижения некоторого критерия



# Подходы к обучению

→ Параллельность по:



# Подходы к обучению

→ Параллельность по:

Модели



# Подходы к обучению

→ Параллельность по:

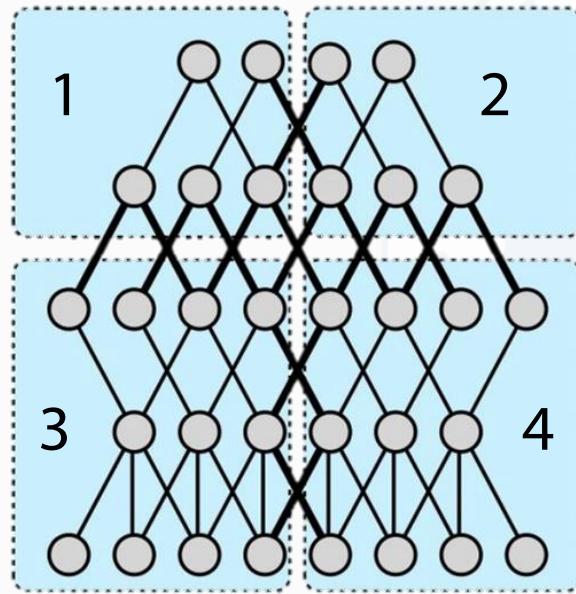
Модели

Данным

# Model Parallel



Зачем:

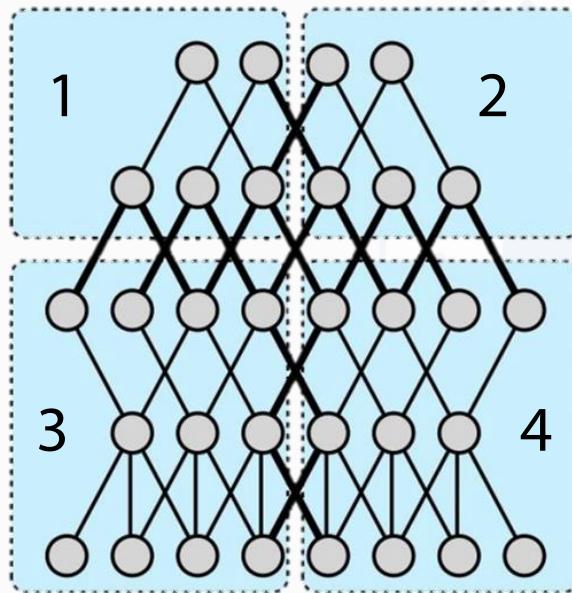


# Model Parallel



Зачем:

- модель большая (workload partitioning)

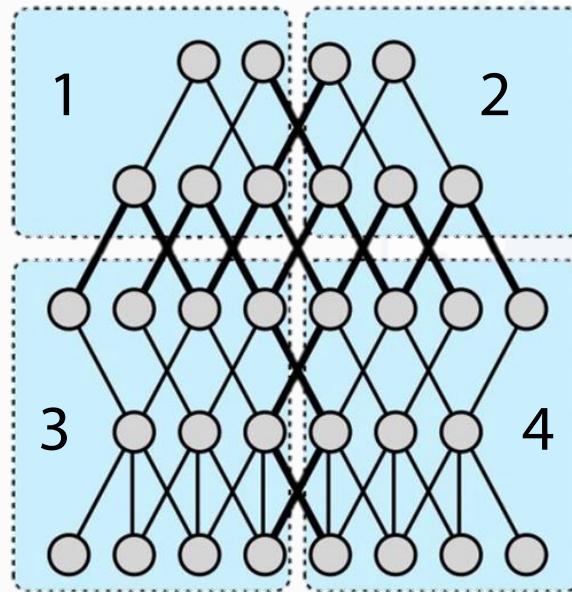


# Model Parallel



Зачем:

- модель большая (workload partitioning)
- есть возможность обучать части независимо



# Model Parallel

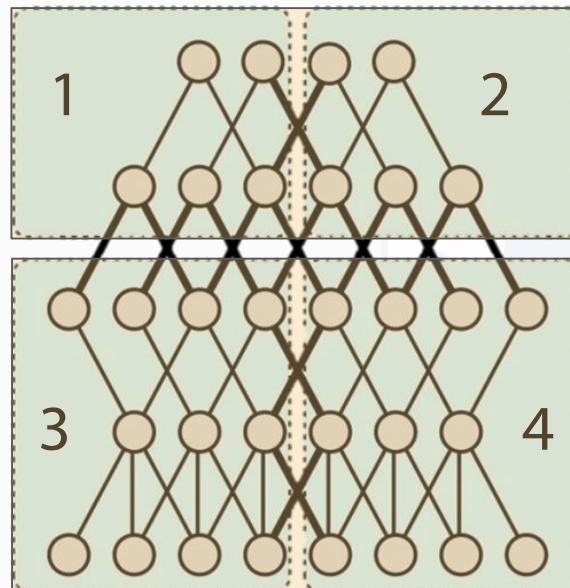


Зачем:

- модель большая (workload partitioning)
- есть возможность обучать части независимо



Можно резать модель вдоль



# Model Parallel

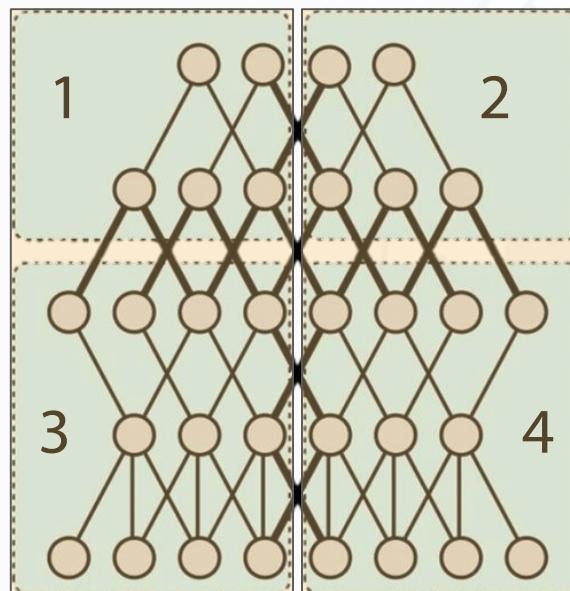


Зачем:

- модель большая (workload partitioning)
- есть возможность обучать части независимо



Можно резать модель вдоль или поперек слоев



# Model Parallel

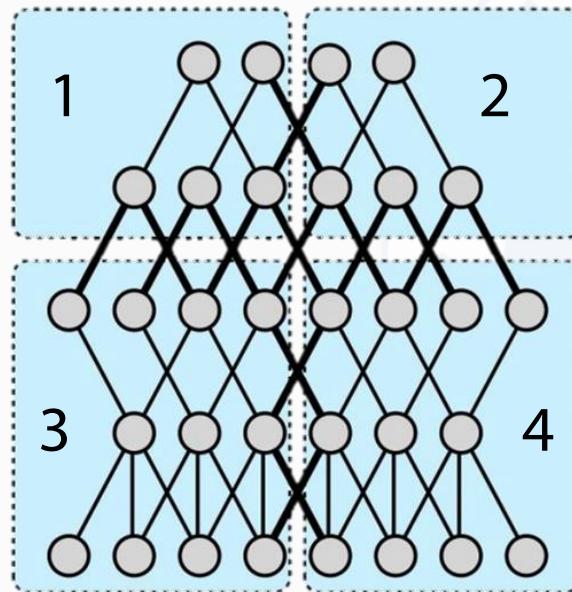


Зачем:

- модель большая (workload partitioning)
- есть возможность обучать части независимо



Можно резать модель вдоль или поперек слоев



# Параллельная обработка данных

→ Подход Data Parallel — разделение обучения по данным

Полный набор данных

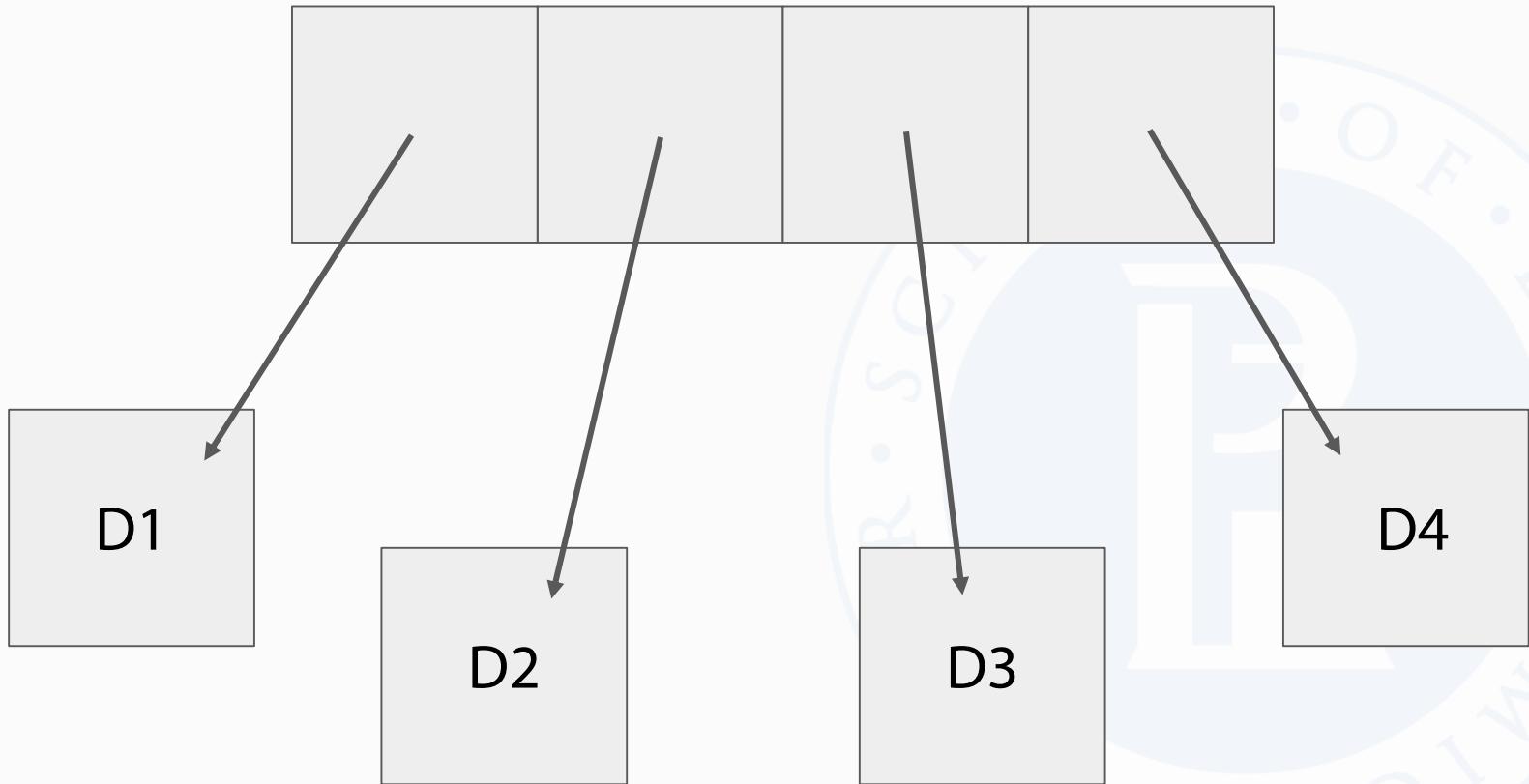


# Параллельная обработка данных



Подход Data Parallel — разделение обучения по данным

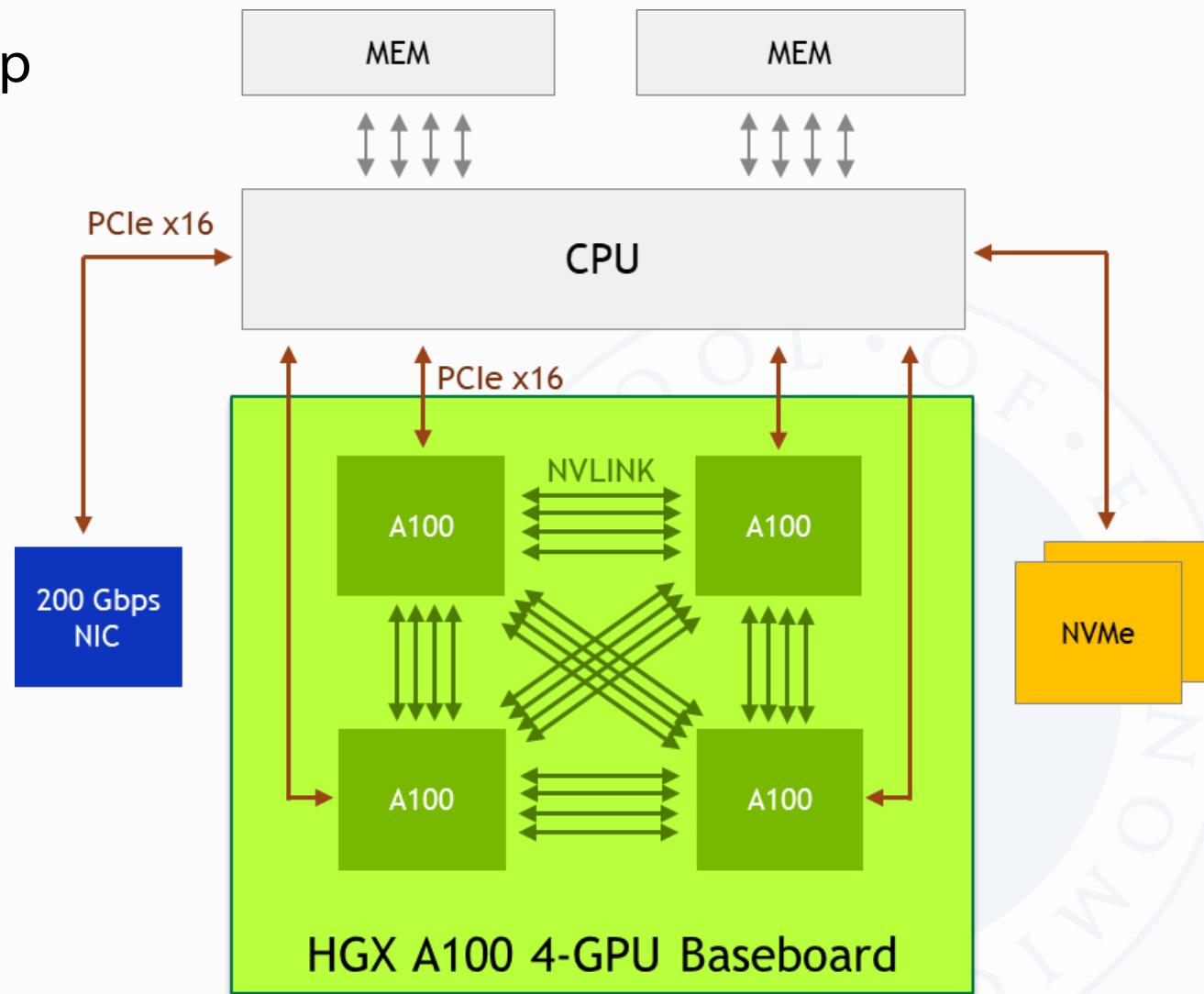
Полный набор данных



# Параллельная обработка данных

→ Простой случай:

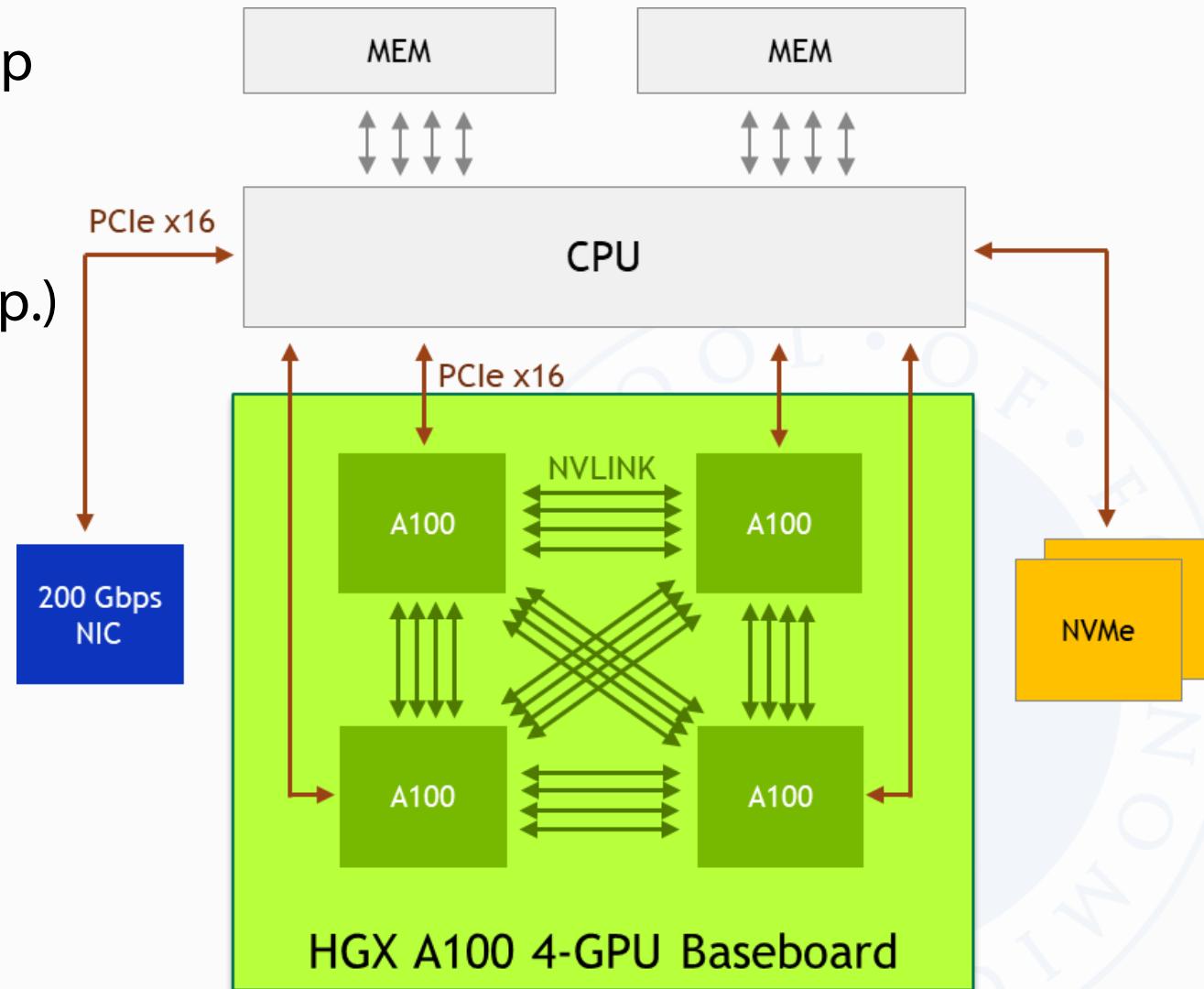
- 1 компьютер



# Параллельная обработка данных

→ Простой случай:

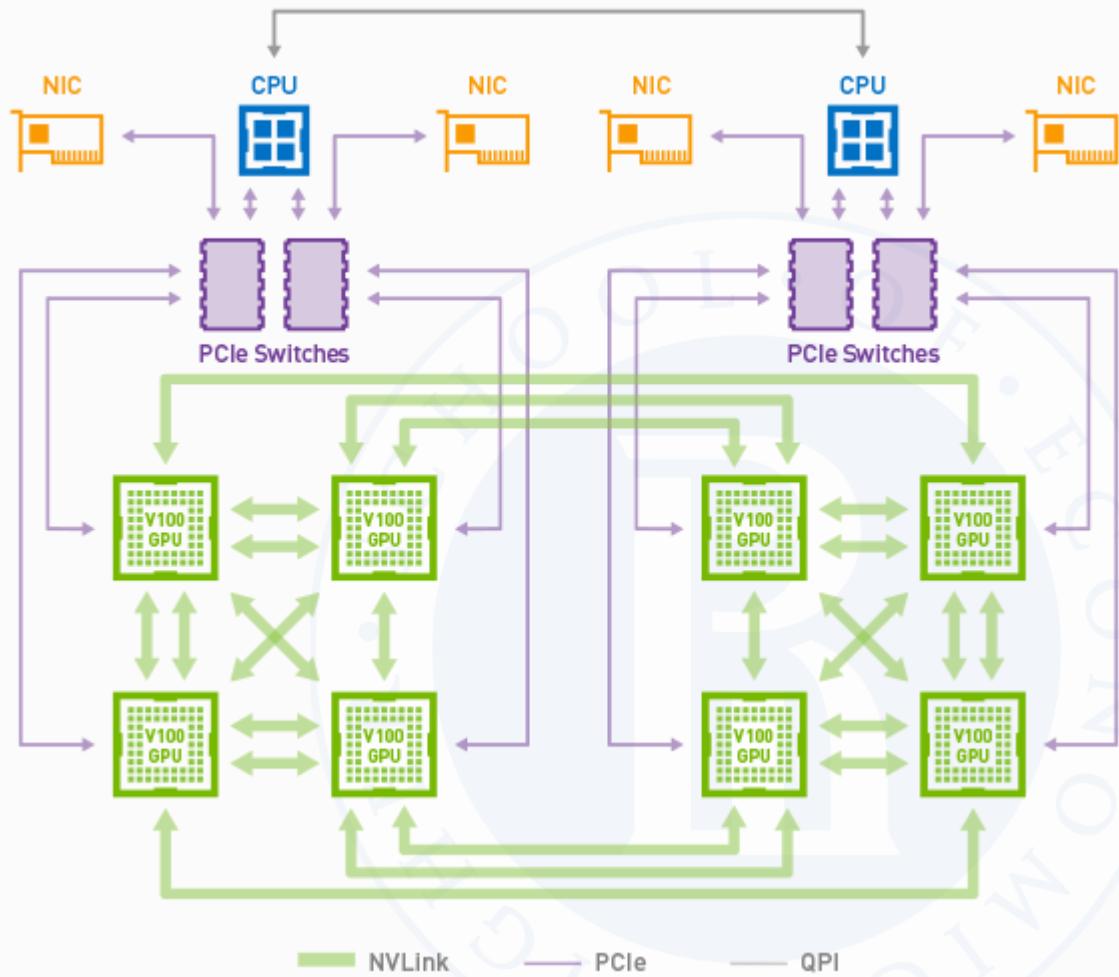
- 1 компьютер
- несколько устройств (GPU/TPU/др.)



# Параллельная обработка данных

→ Сложный случай:

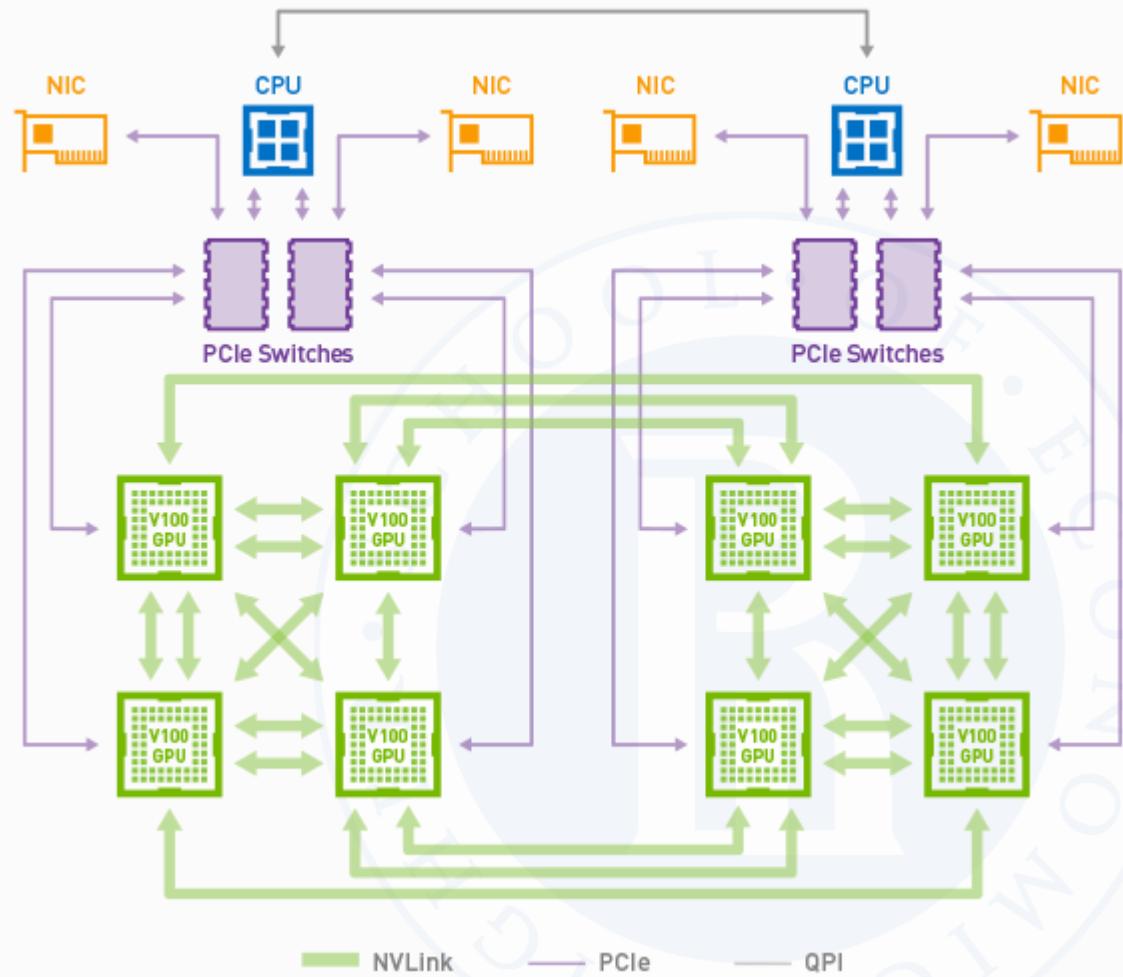
- кластер с одним устройством на каждом узле



# Параллельная обработка данных

→ Сложный случай:

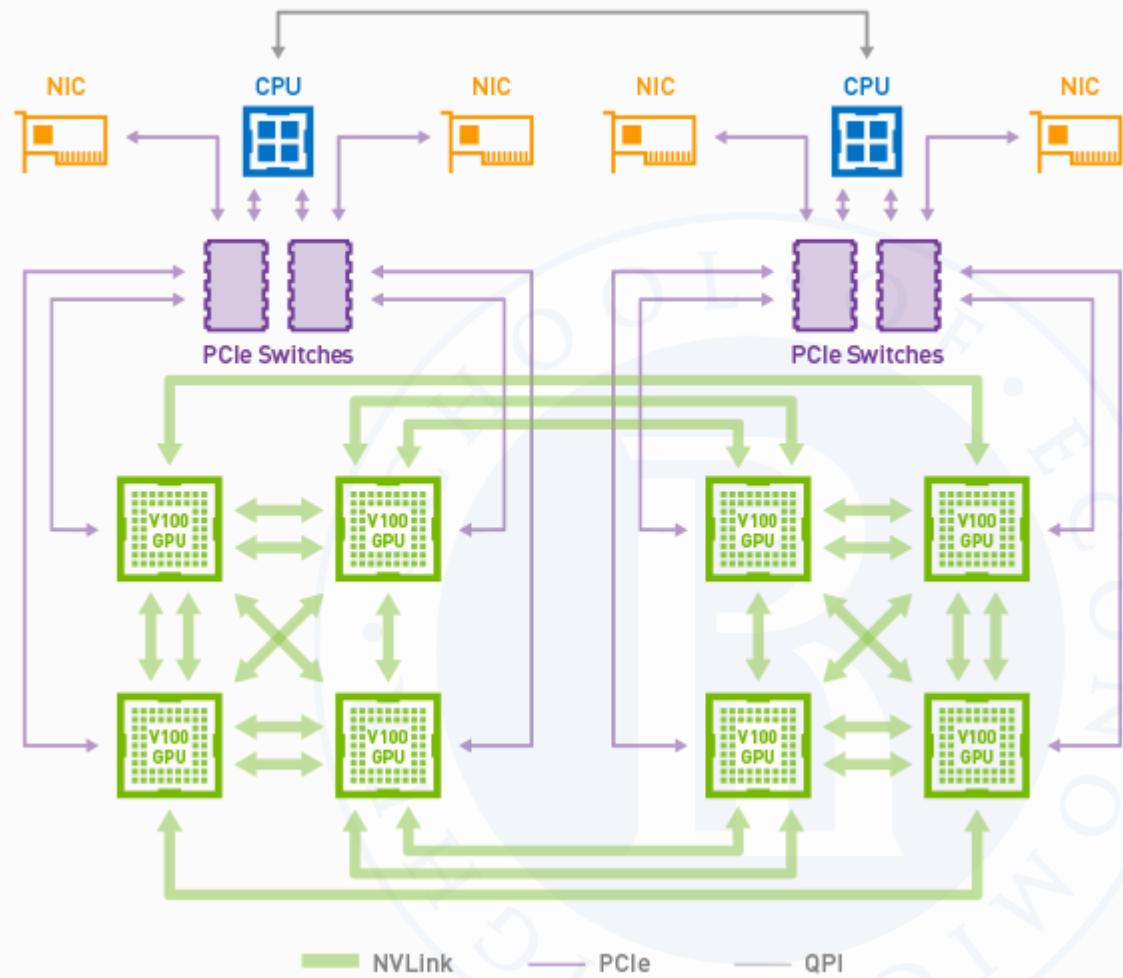
- кластер с одним устройством на каждом узле
- кластер с несколькими устройствами на каждом узле



# Параллельная обработка данных

→ Сложный случай:

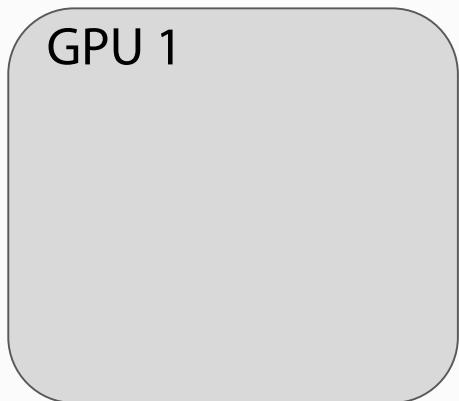
- кластер с одним устройством на каждом узле
- кластер с несколькими устройствами на каждом узле
- в следующем видео



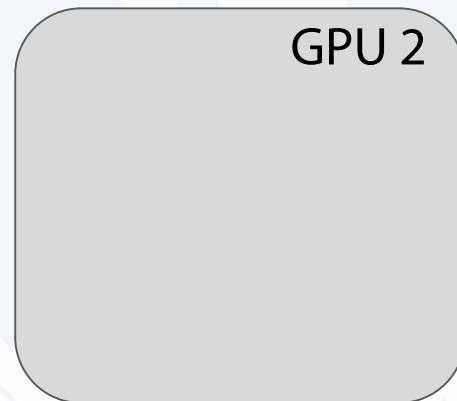
# Параллельная обработка данных



У нас есть устройства — обработчики



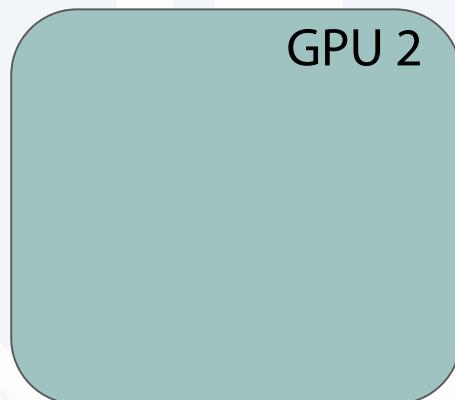
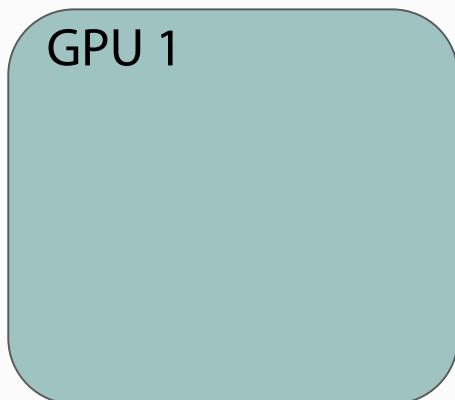
GPU 1



GPU 2

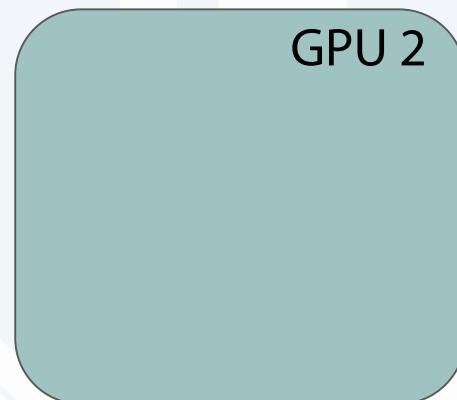
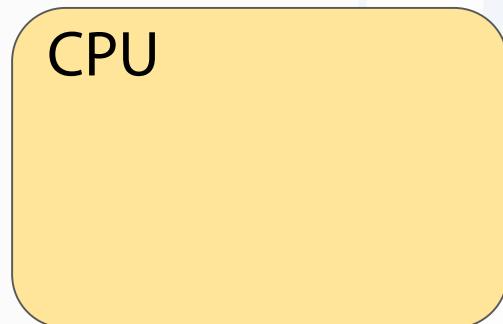
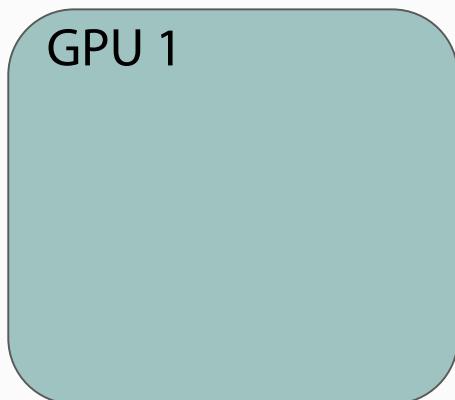
# Параллельная обработка данных

- У нас есть устройства — обработчики
- По ним мы делим данные для обработки



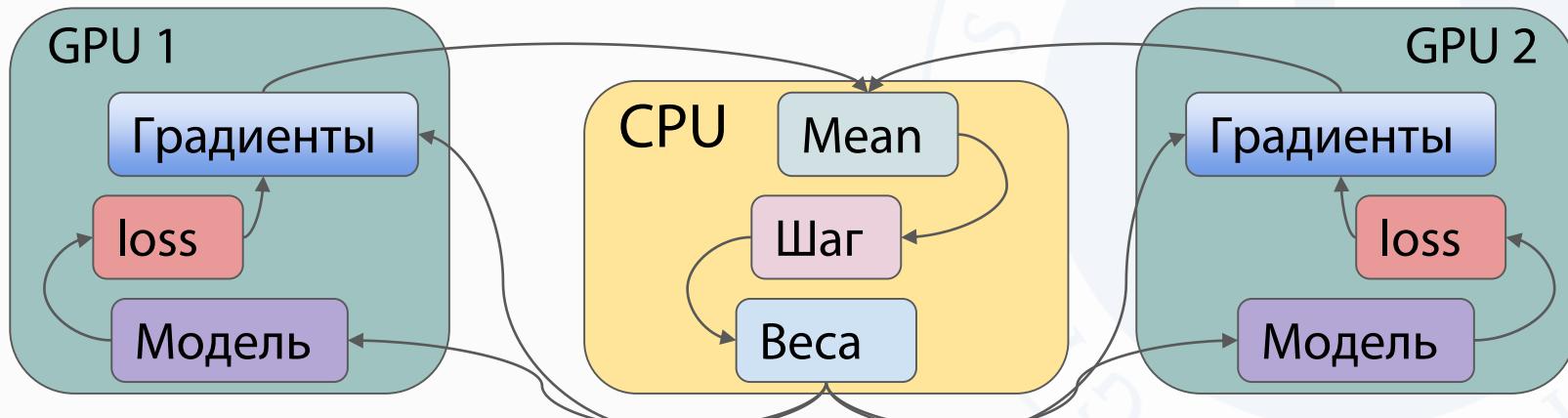
# Параллельная обработка данных

- У нас есть устройства — обработчики
- По ним мы делим данные для обработки
- Есть выделенный координатор



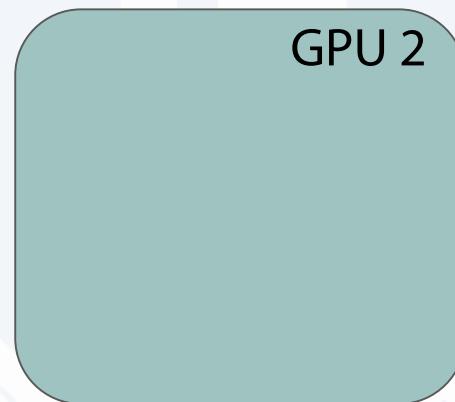
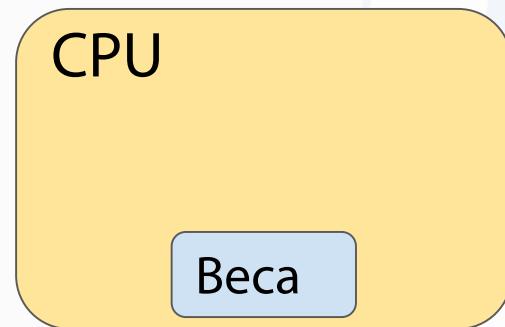
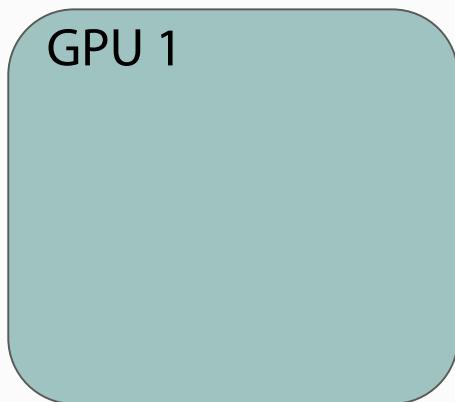
# Параллельная обработка данных

- У нас есть устройства — обработчики
- По ним мы делим данные для обработки
- Есть выделенный координатор
- Изменение весов происходит в координаторе, но сами шаги вычисляются параллельно на устройствах



# Параллельная обработка данных

→ Простой случай

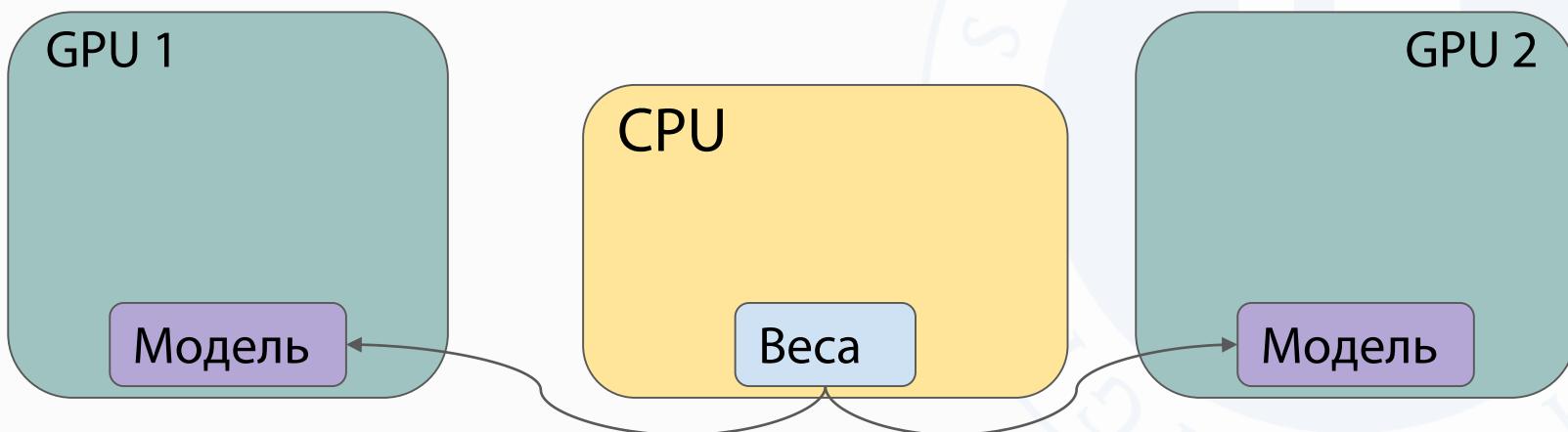


# Параллельная обработка данных



Простой случай

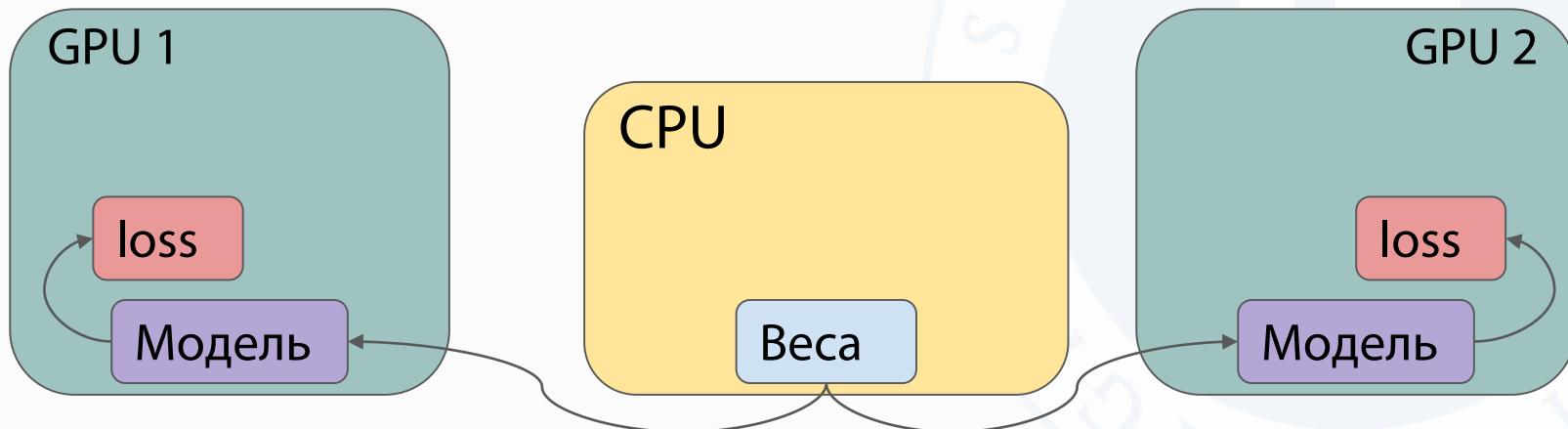
- Рассылаем копии модели на GPU



# Параллельная обработка данных

## → Простой случай

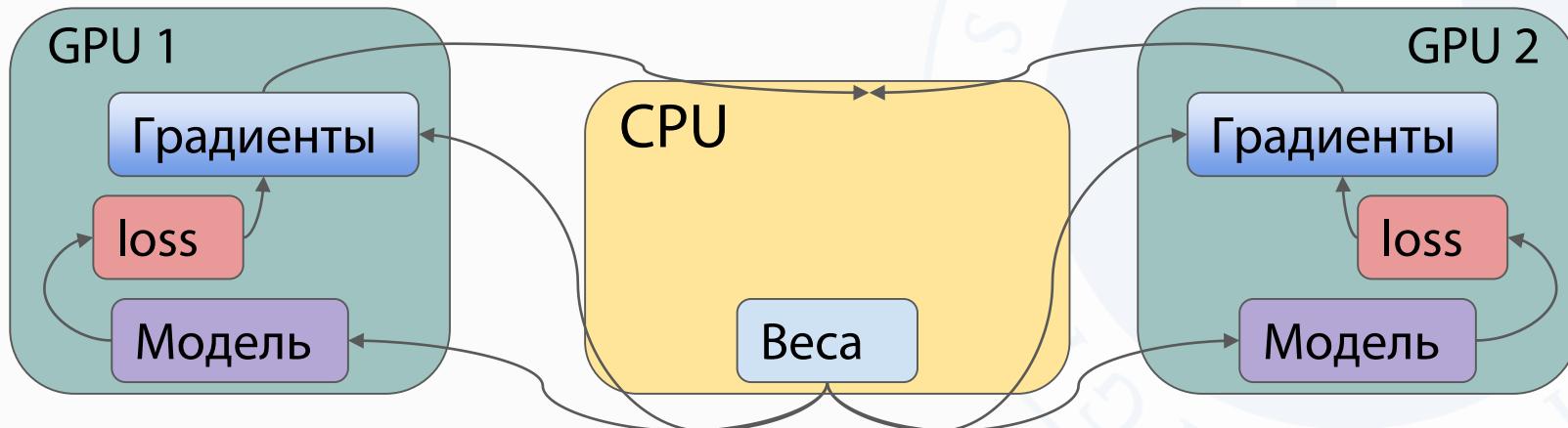
- Рассылаем копии модели на GPU
- Применяем модель на части батча



# Параллельная обработка данных

## → Простой случай

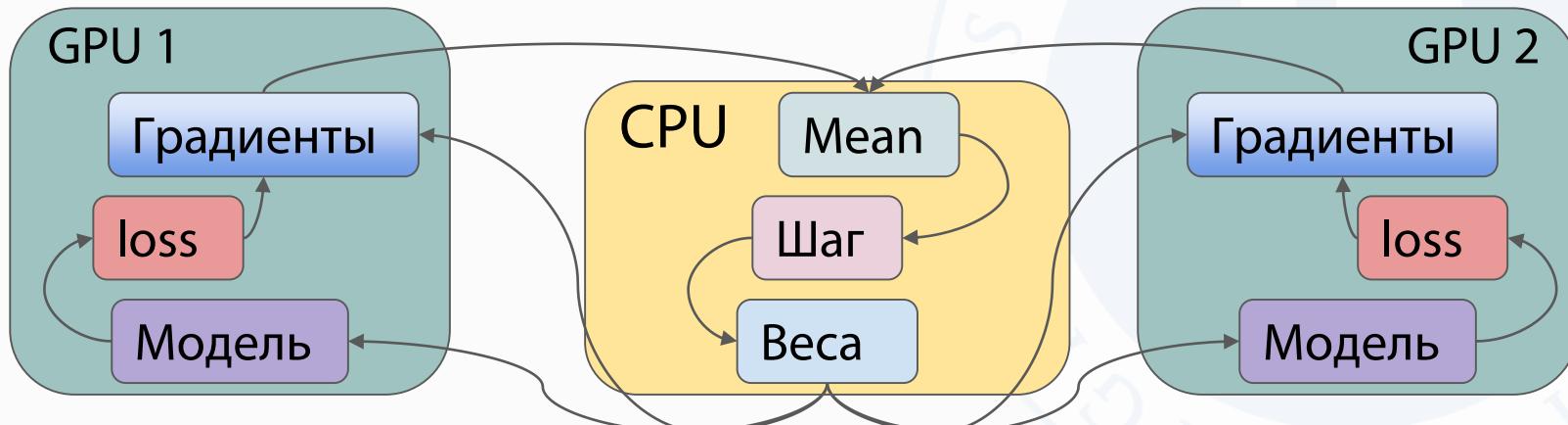
- Рассылаем копии модели на GPU
- Применяем модель на части батча
- Вклады в градиент отправляем координатору



# Параллельная обработка данных

## → Простой случай

- Рассылаем копии модели на GPU
- Применяем модель на части батча
- Вклады в градиент отправляем координатору
- Суммируем и заново рассылаем



# Протокол Message Passing Interface

Процессы:

0

1

2

# Протокол Message Passing Interface

Процессы:

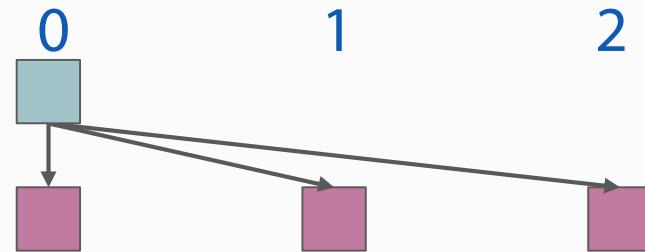
→ Отправить всем  
MPI\_BCAST



# Протокол Message Passing Interface

Процессы:

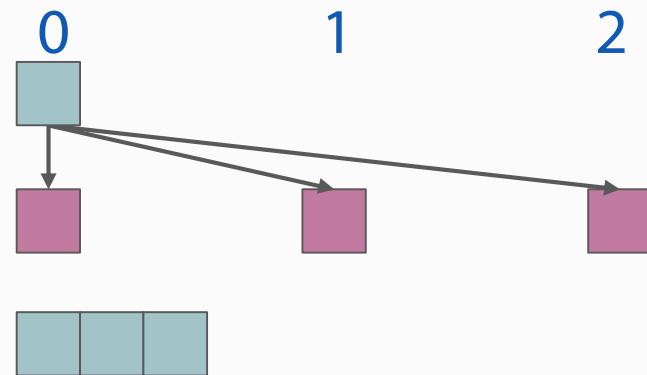
- Отправить всем  
MPI\_BCAST



# Протокол Message Passing Interface

Процессы:

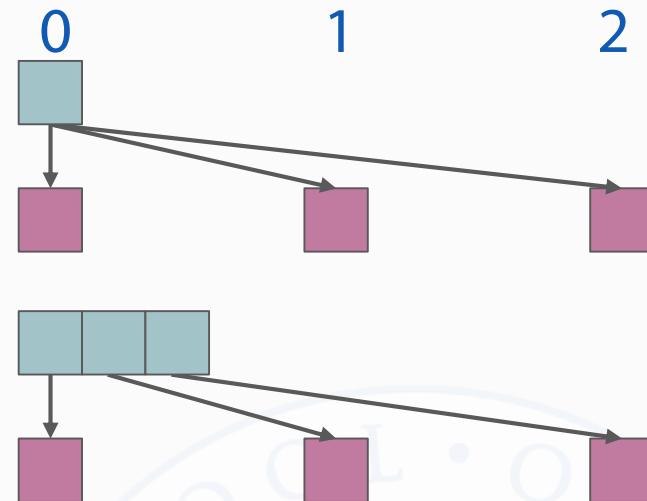
- Отправить всем  
MPI\_BCAST



# Протокол Message Passing Interface

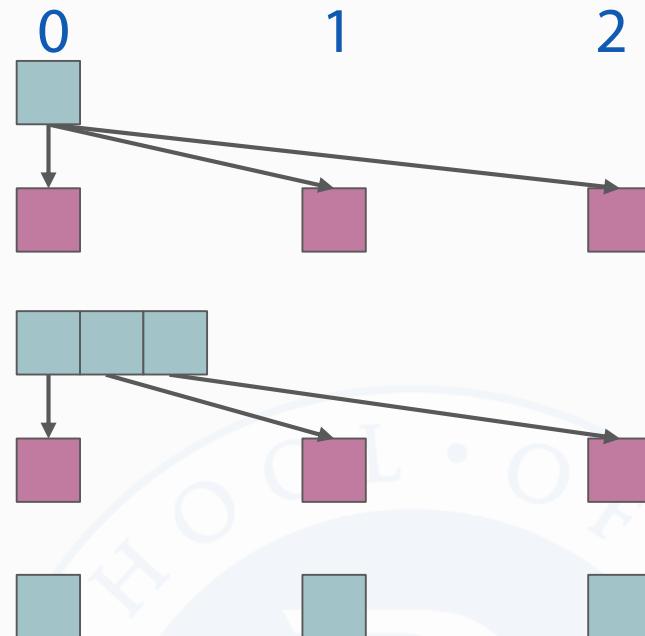
Процессы:

- Отправить всем MPI\_BCAST
- Распределить MPI\_SCATTER



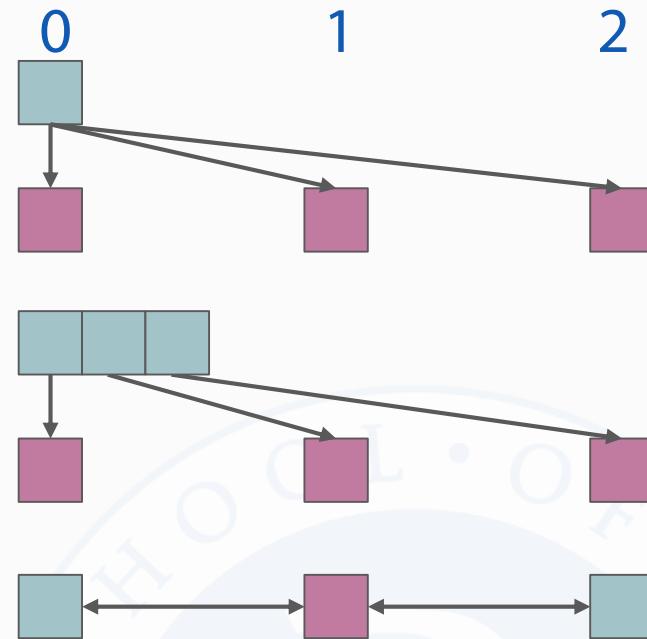
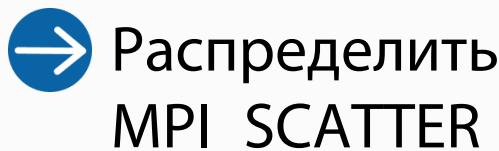
# Протокол Message Passing Interface

Процессы:



# Протокол Message Passing Interface

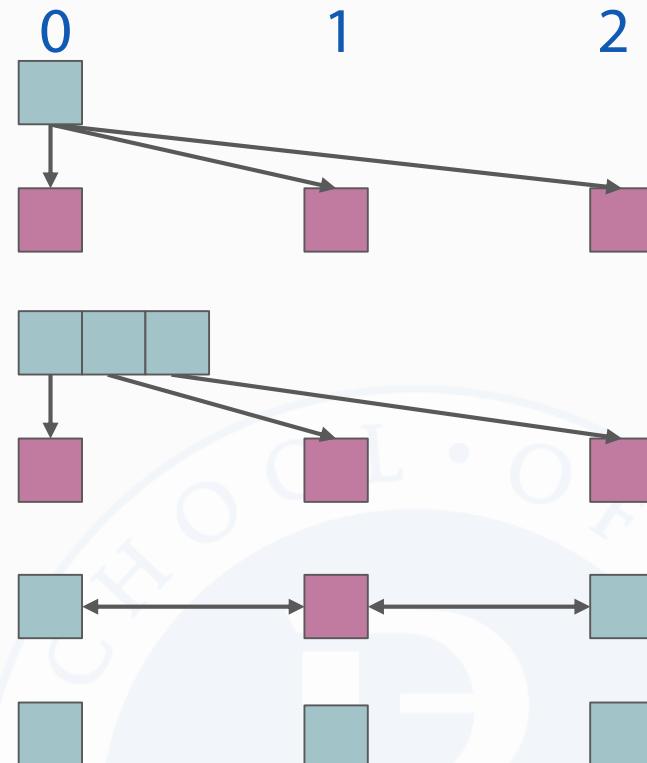
Процессы:



# Протокол Message Passing Interface

Процессы:

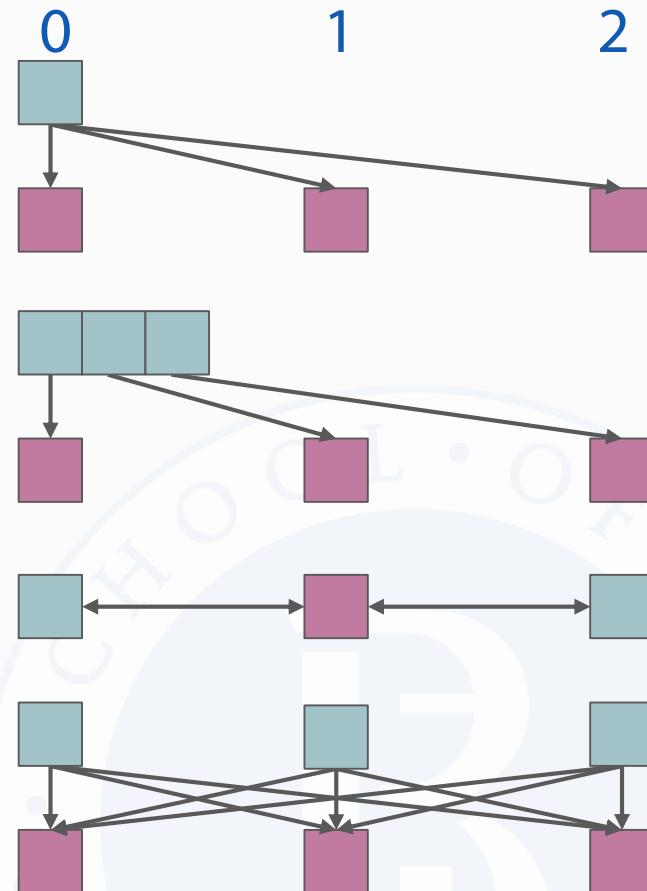
- Отправить всем MPI\_BCAST
- Распределить MPI\_SCATTER
- Отправить/Получить MPI\_SEND/MPI\_RECV
- Обработать блоки [для всех] MPI\_REDUCE[ALL]



# Протокол Message Passing Interface

Процессы:

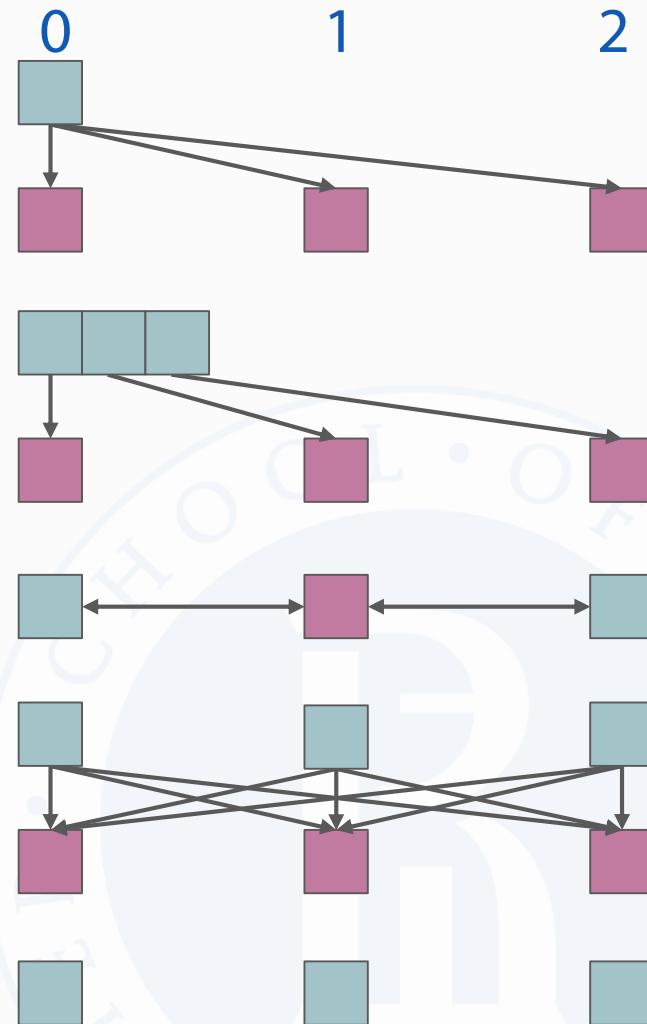
- Отправить всем MPI\_BCAST
- Распределить MPI\_SCATTER
- Отправить/Получить MPI\_SEND/MPI\_RECV
- Обработать блоки [для всех] MPI\_REDUCE[ALL]



# Протокол Message Passing Interface

Процессы:

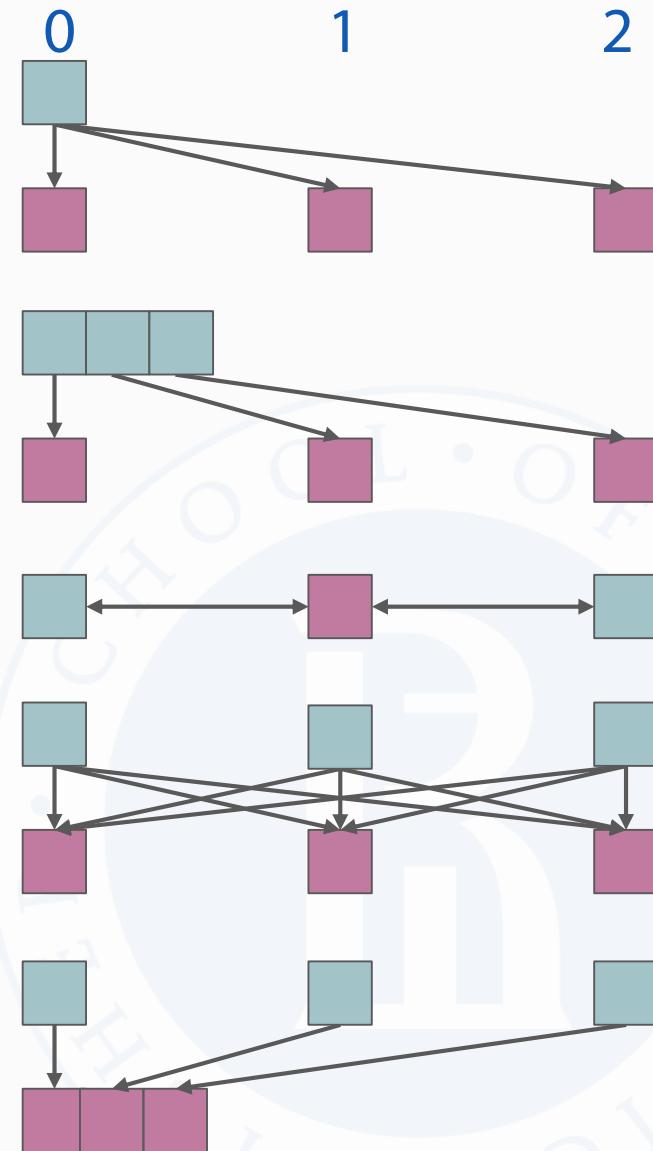
- Отправить всем MPI\_BCAST
- Распределить MPI\_SCATTER
- Отправить/Получить MPI\_SEND/MPI\_RECV
- Обработать блоки [для всех] MPI\_REDUCE[ALL]
- Собрать MPI\_GATHER



# Протокол Message Passing Interface

Процессы:

- Отправить всем MPI\_BCAST
- Распределить MPI\_SCATTER
- Отправить/Получить MPI\_SEND/MPI\_RECV
- Обработать блоки [для всех] MPI\_REDUCE[ALL]
- Собрать MPI\_GATHER



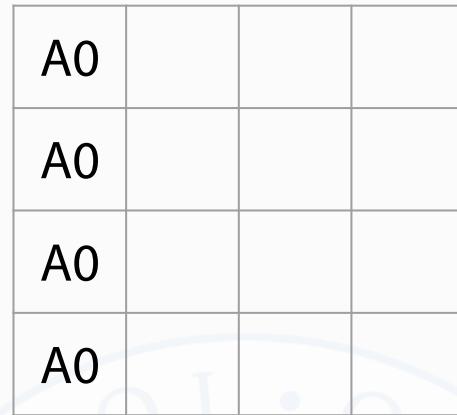
# **MPI. Один ко многим и многое к одному**



ОДИН КО МНОГИМ



**MPI\_BCAST**



# MPI. Один ко многим и многое к одному

Данные

A0			

один ко многим



`MPI_BCAST`

A0			

A0			
A1			
A2			
A3			

все к одному



A0	A1	A2	A3

# **MPI. Один ко многим и многое к одному**

Данные

Процессы	A0	A1	A2	A3

ОДИН КО МНОГИМ



**MPI\_SCATTER**

A0			
A1			
A2			
A3			

# MPI. Много ко многим

Данные

Процессы

A0			
A1			
A2			
A3			

все ко всем

ALLGATHER

A0	A1	A2	A3
A0	A1	A2	A3
A0	A1	A2	A3
A0	A1	A2	A3

# MPI. Много ко многим

Данные

Процессы

A0			
A1			
A2			
A3			

все ко всем

ALLGATHER

A0	A1	A2	A3
A0	A1	A2	A3
A0	A1	A2	A3
A0	A1	A2	A3

A0			
A1			
A2			
A3			

все ко всем

ALLREDUCE  
с функцией F

F(A0, A1, A2, A3)

# MPI. Пример работы REDUCE и ALLREDUCE

Процессы:

0

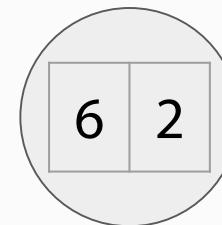
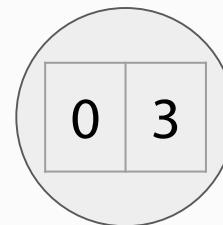
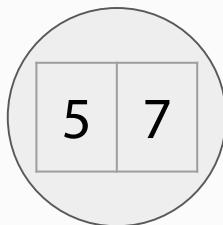
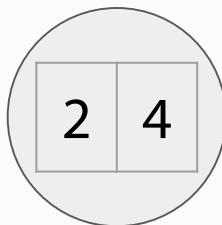
1

2

3



Начальные  
данные:



# MPI. Пример работы REDUCE и ALLREDUCE

Процессы:

0

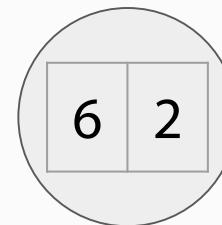
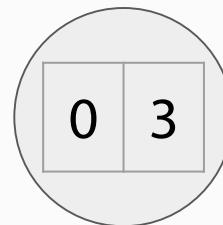
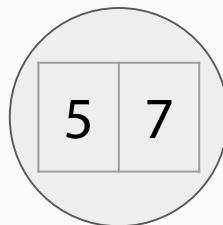
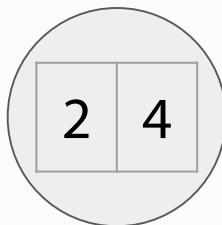
1

2

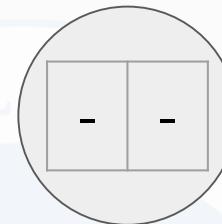
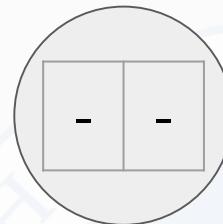
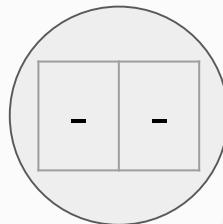
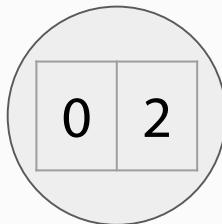
3



Начальные  
данные:



MPI\_REDUCE  
с MPI\_MIN,  
корень = 0



# MPI. Пример работы REDUCE и ALLREDUCE

Процессы:

0

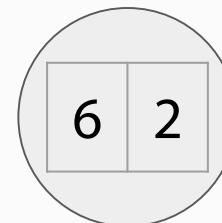
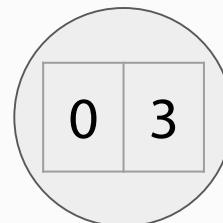
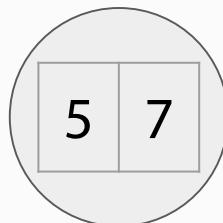
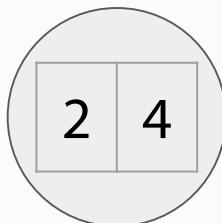
1

2

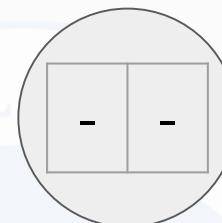
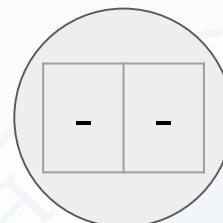
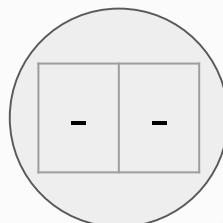
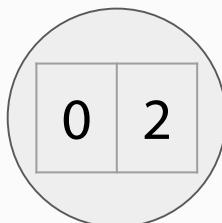
3



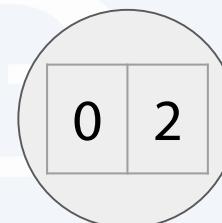
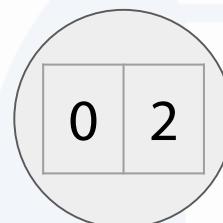
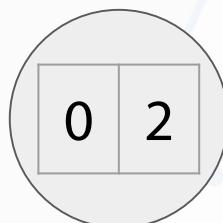
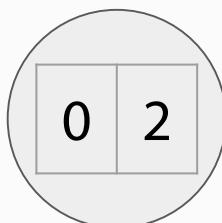
Начальные  
данные:



MPI\_REDUCE  
с MPI\_MIN,  
корень = 0



MPI\_ALLREDUCE  
с MPI\_MIN



# MPI. Пример работы REDUCE и ALLREDUCE

Процессы:

0

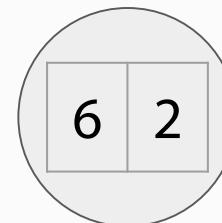
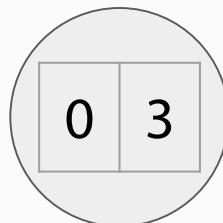
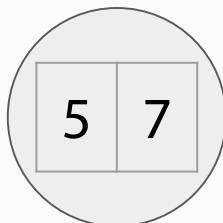
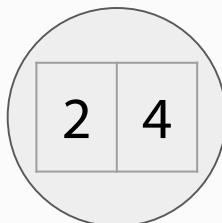
1

2

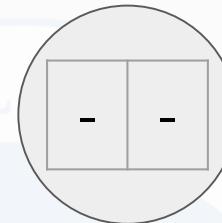
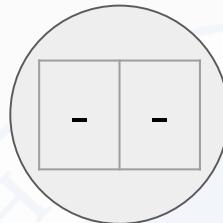
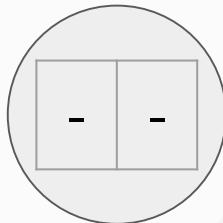
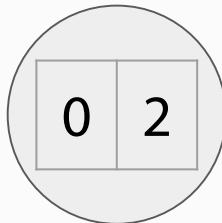
3



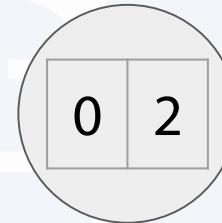
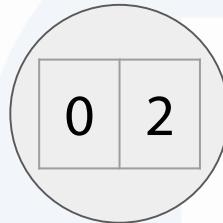
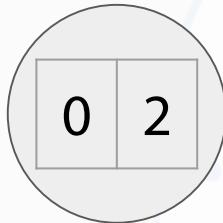
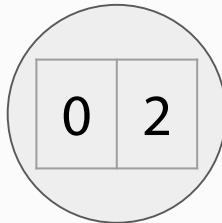
Начальные  
данные:



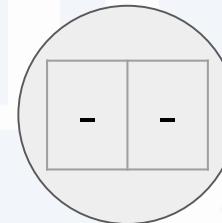
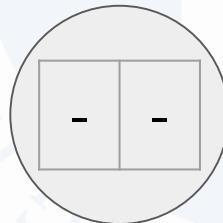
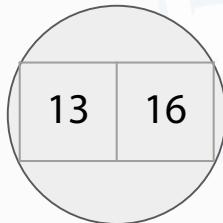
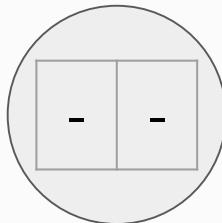
MPI\_REDUCE  
с MPI\_MIN,  
корень = 0



MPI\_ALLREDUCE  
с MPI\_MIN

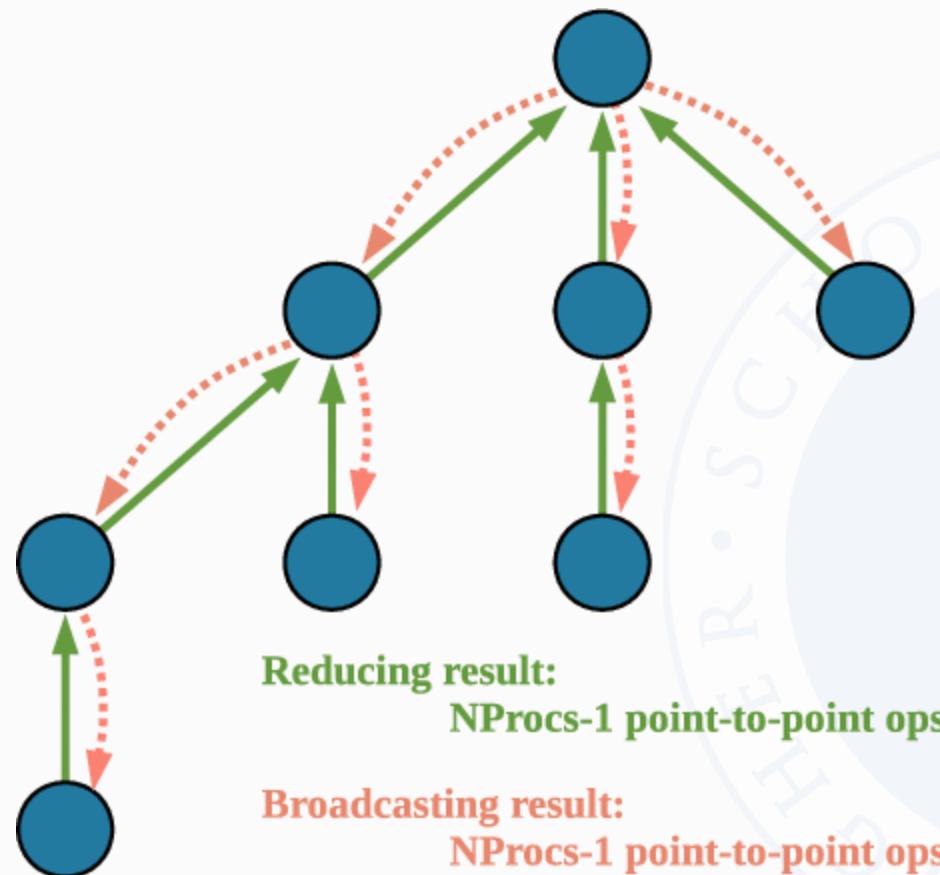


MPI\_REDUCE  
с MPI\_SUM,  
корень = 1



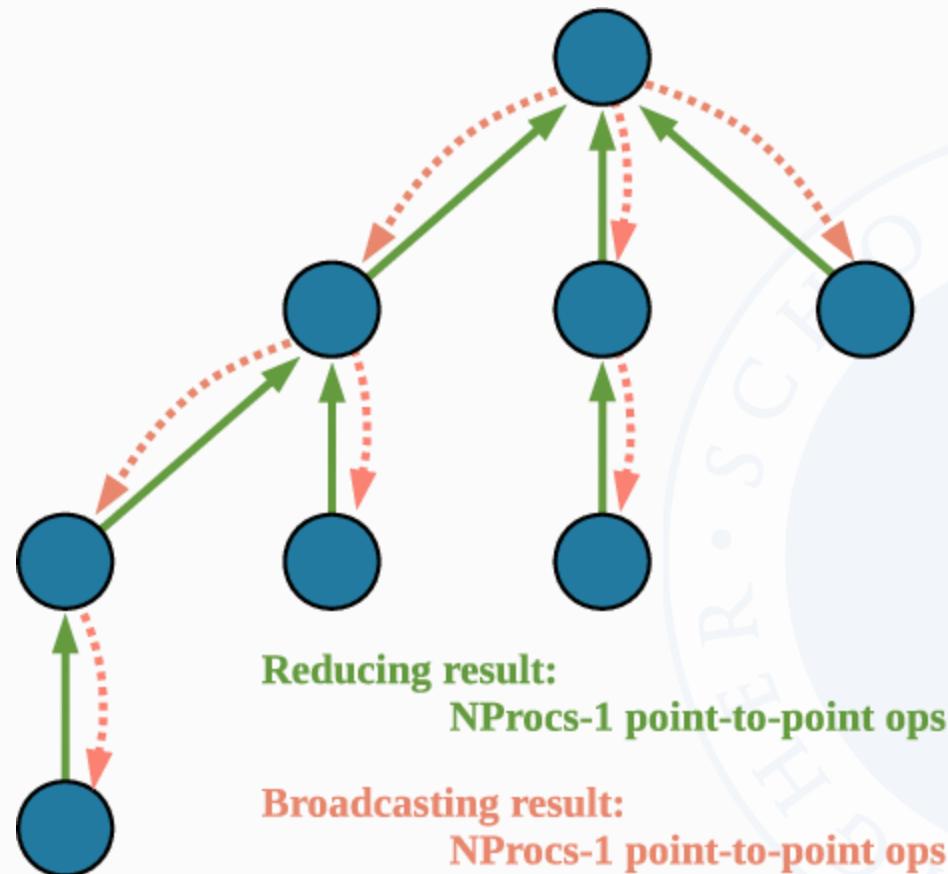
# Tree-Allreduce

→ Встречался при использовании Vowpal Wabbit для оптимизации передачи данных по сети



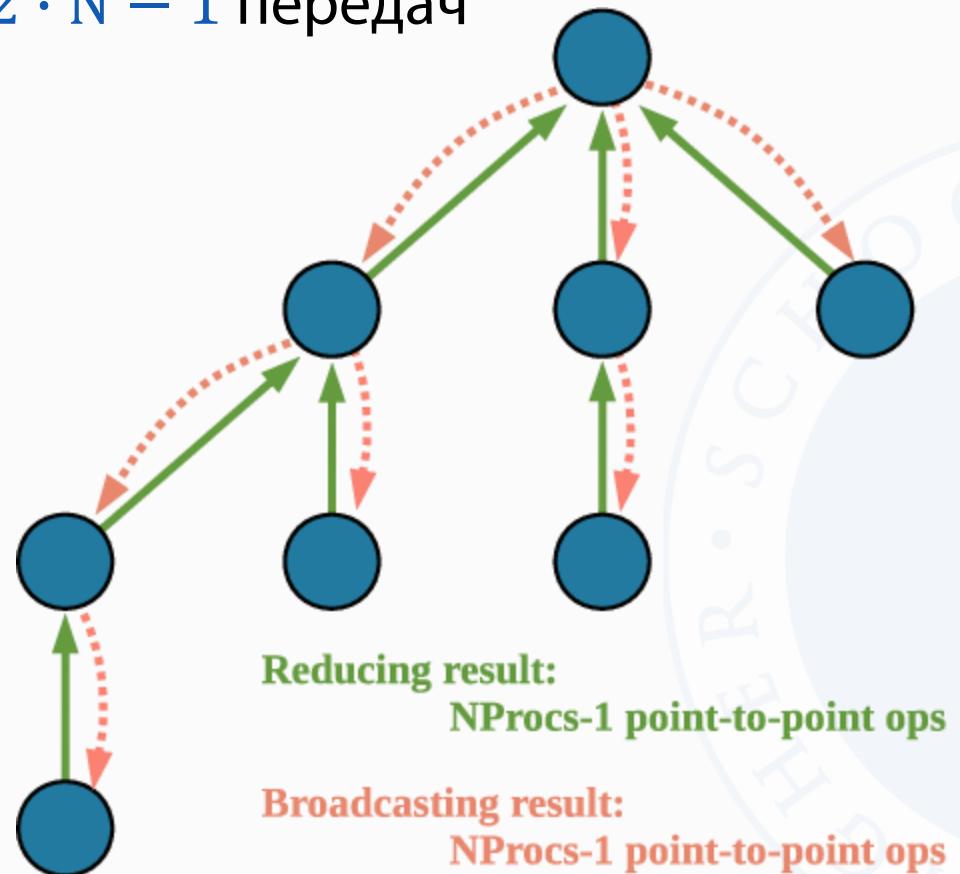
# Tree-Allreduce

- Сначала происходит reduce в корень — сбор данных
- Затем — broadcast из корня до всех — обновление данных



# Tree-Allreduce

- Оптимальная утилизация сети за счет преимуществ топологии: не прямые передачи, а по ребрам дерева
- Всего  $2 \cdot N - 1$  передач



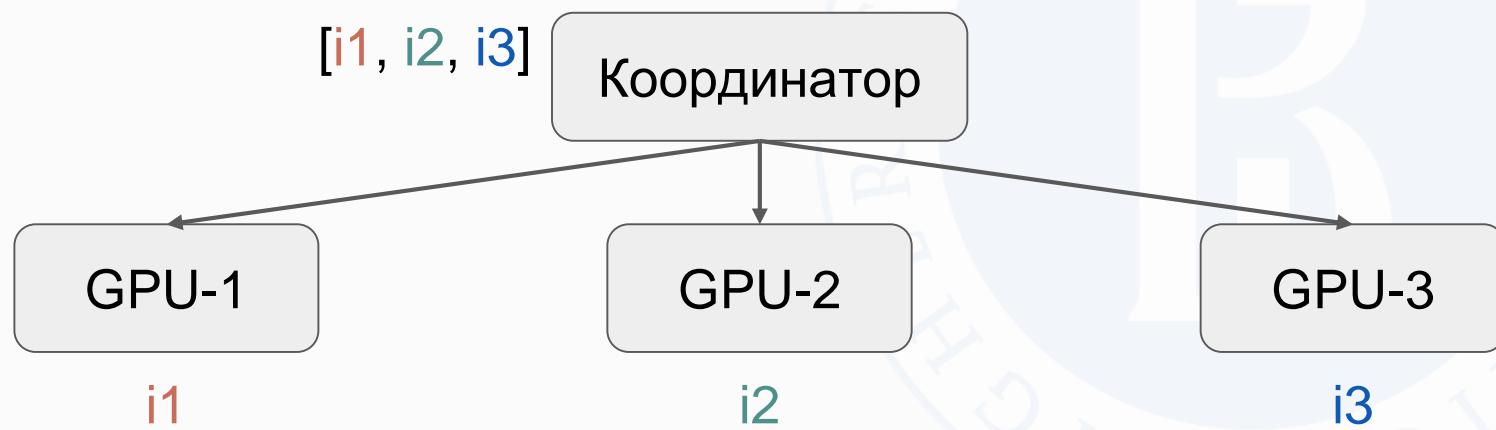
# Процесс обучения

→ Обучение в терминах MPI



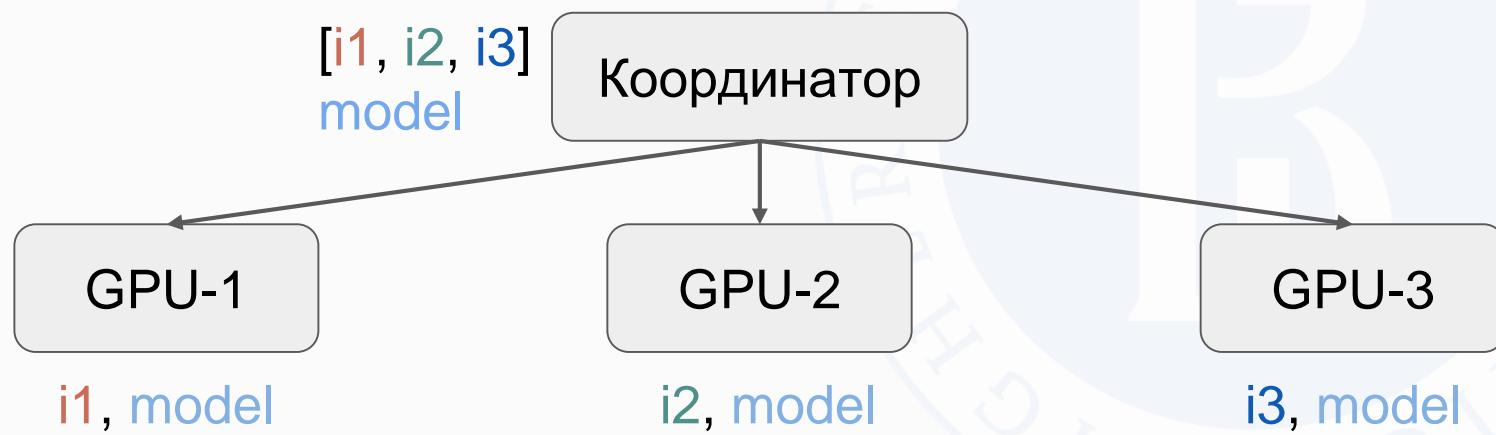
# Процесс обучения

## 1. Scatter блоков данных на все GPU



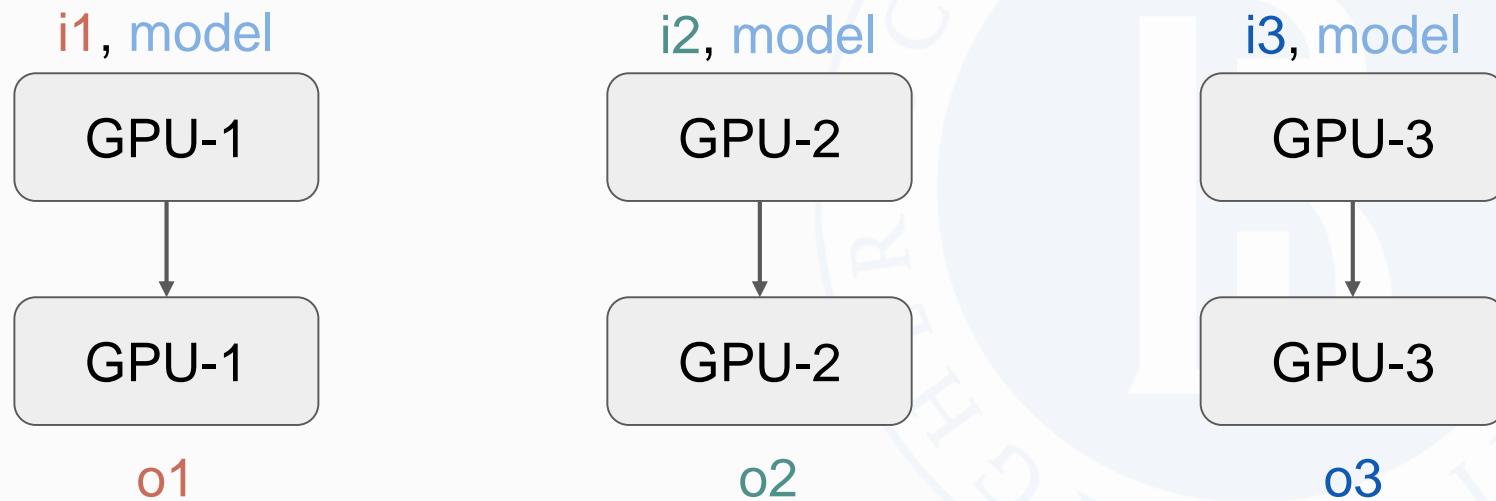
# Процесс обучения

1. Scatter блоков данных на все GPU
2. Replicate модель на все устройства



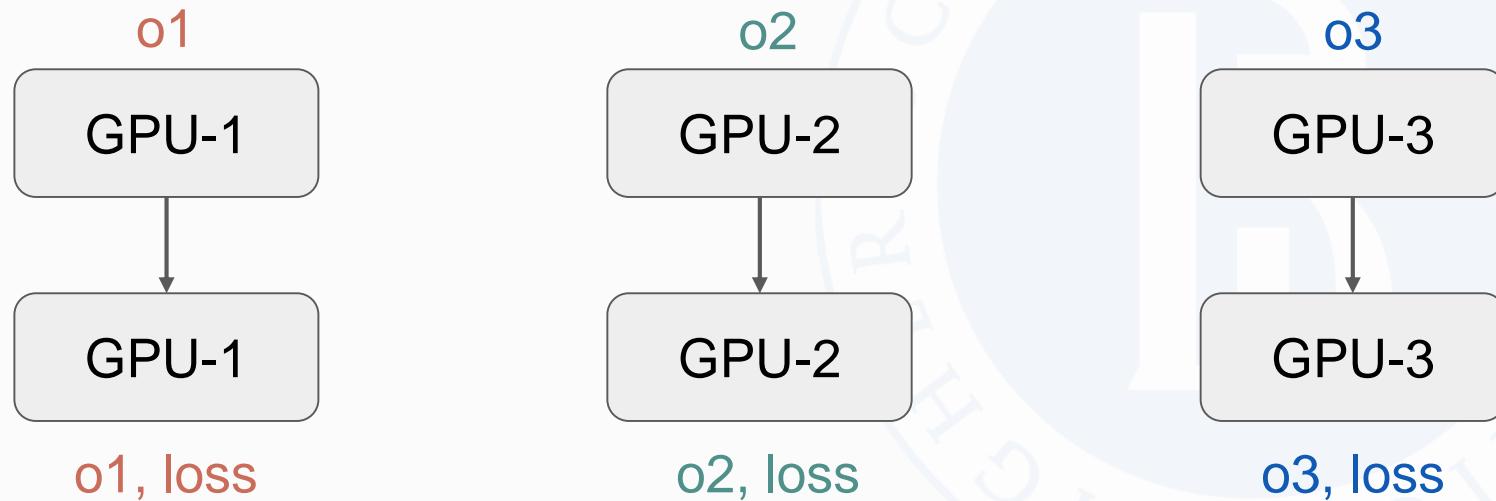
# Процесс обучения

1. Scatter блоков данных на все GPU
2. Replicate модель на все устройства
3. Forward шаг по данным на каждом устройстве



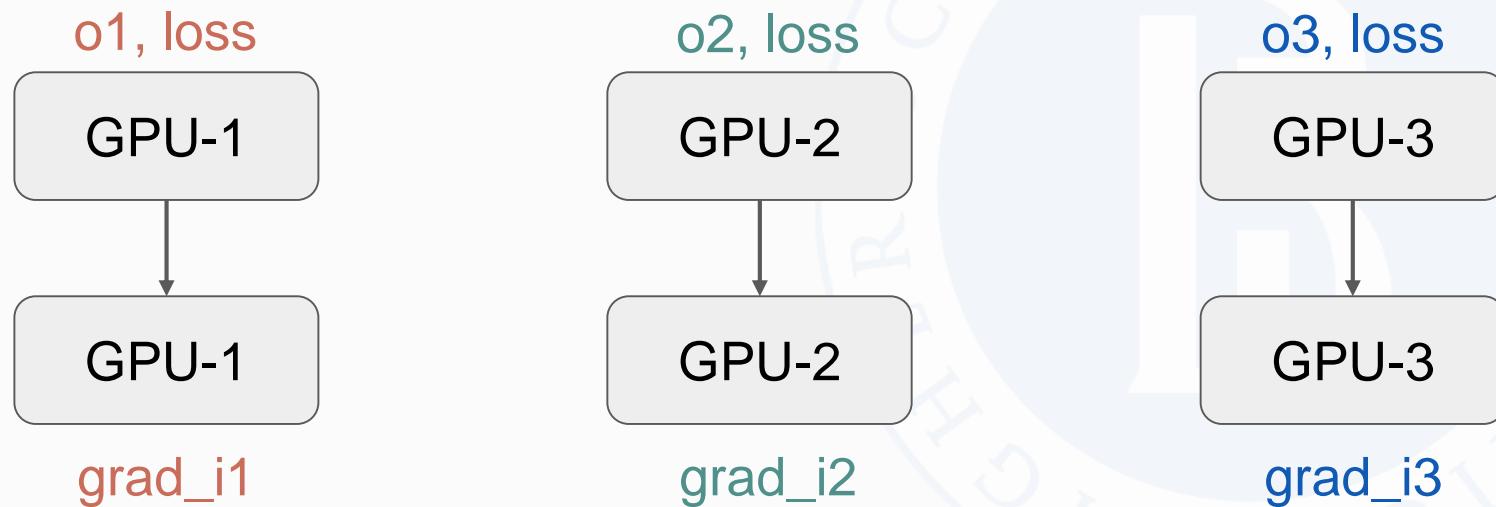
# Процесс обучения

1. Scatter блоков данных на все GPU
2. Replicate модель на все устройства
3. Forward шаг по данным на каждом устройстве
4. Подсчет loss на каждом устройстве



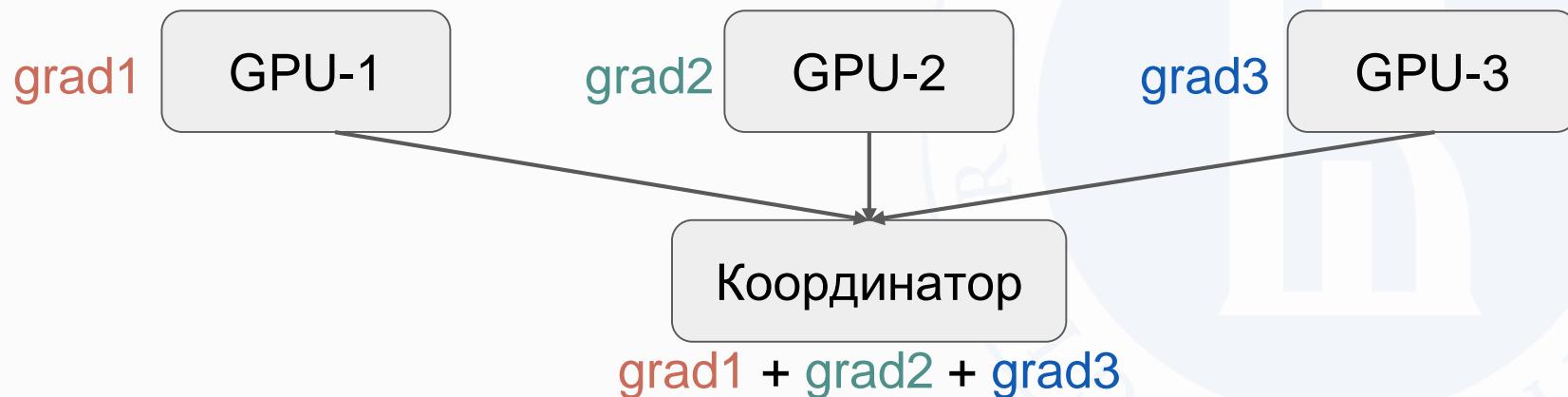
# Процесс обучения

1. Scatter блоков данных на все GPU
2. Replicate модель на все устройства
3. Forward шаг по данным на каждом устройстве
4. Подсчет loss на каждом устройстве
5. Backward проход по данным, пересчет градиентов



# Процесс обучения

1. Scatter блоков данных на все GPU
2. Replicate модель на все устройства
3. Forward шаг по данным на каждом устройстве
4. Подсчет loss на каждом устройстве
5. Backward проход по данным, пересчет градиентов
6. Reduce градиентов в исходное устройство (CPU)



# Итоги



В этом видео мы познакомились с:



# Итоги



В этом видео мы познакомились с:



подходом Model Parallel



# Итоги



В этом видео мы познакомились с:



подходом Model Parallel



параллелизмом по данным на одном узле

# Итоги



В этом видео мы познакомились с:

- подходом Model Parallel
- параллелизмом по данным на одном узле
- протоколом Message Passing Interface

# Итоги



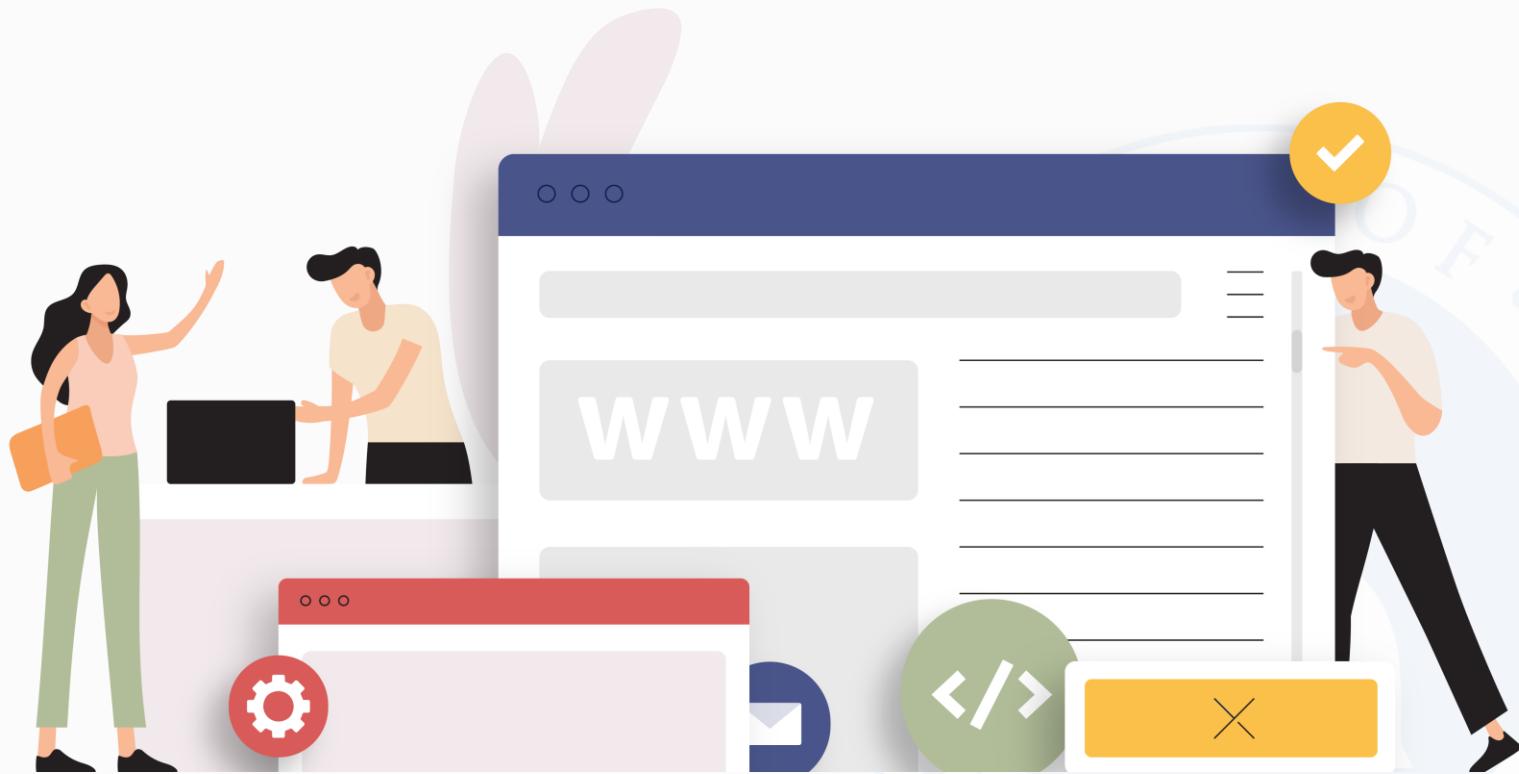
В этом видео мы познакомились с:

- подходом Model Parallel
- параллелизмом по данным на одном узле
- протоколом Message Passing Interface
- реализацией алгоритма параллельного обучения по данным в терминах MPI

# Далее



В следующем видео мы поговорим о параллельности обучения нейронных сетей на нескольких узлах и возможных проблемах при таком подходе



# **Обучение на кластере**



# План



Параллельность по данным на кластере



# План



Параллельность по данным на кластере



Parameter Server

# План



Параллельность по данным на кластере



Parameter Server



Асинхронный SGD



# План



Параллельность по данным на кластере



Parameter Server



Асинхронный SGD



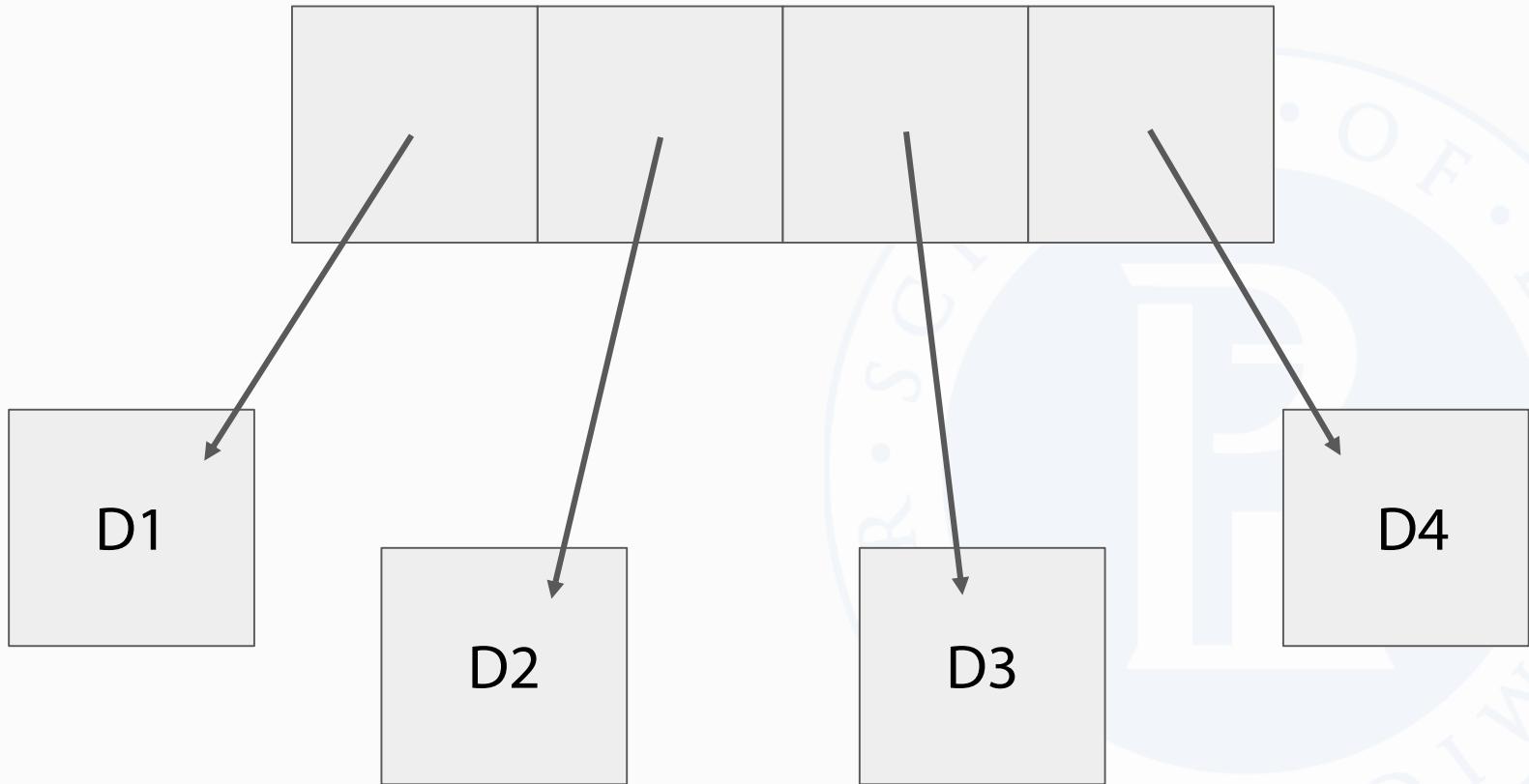
Проблемы обучения параллельного по данным

# Параллельная обработка данных



Подход Data Parallel — разделение обучения по данным

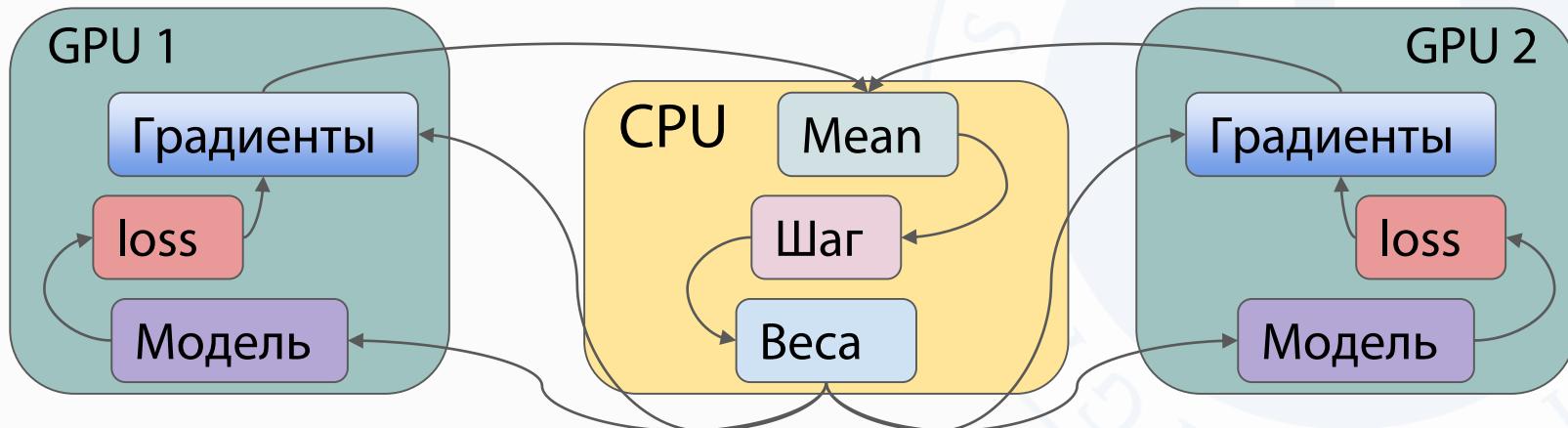
Полный набор данных



# Параллельная обработка данных

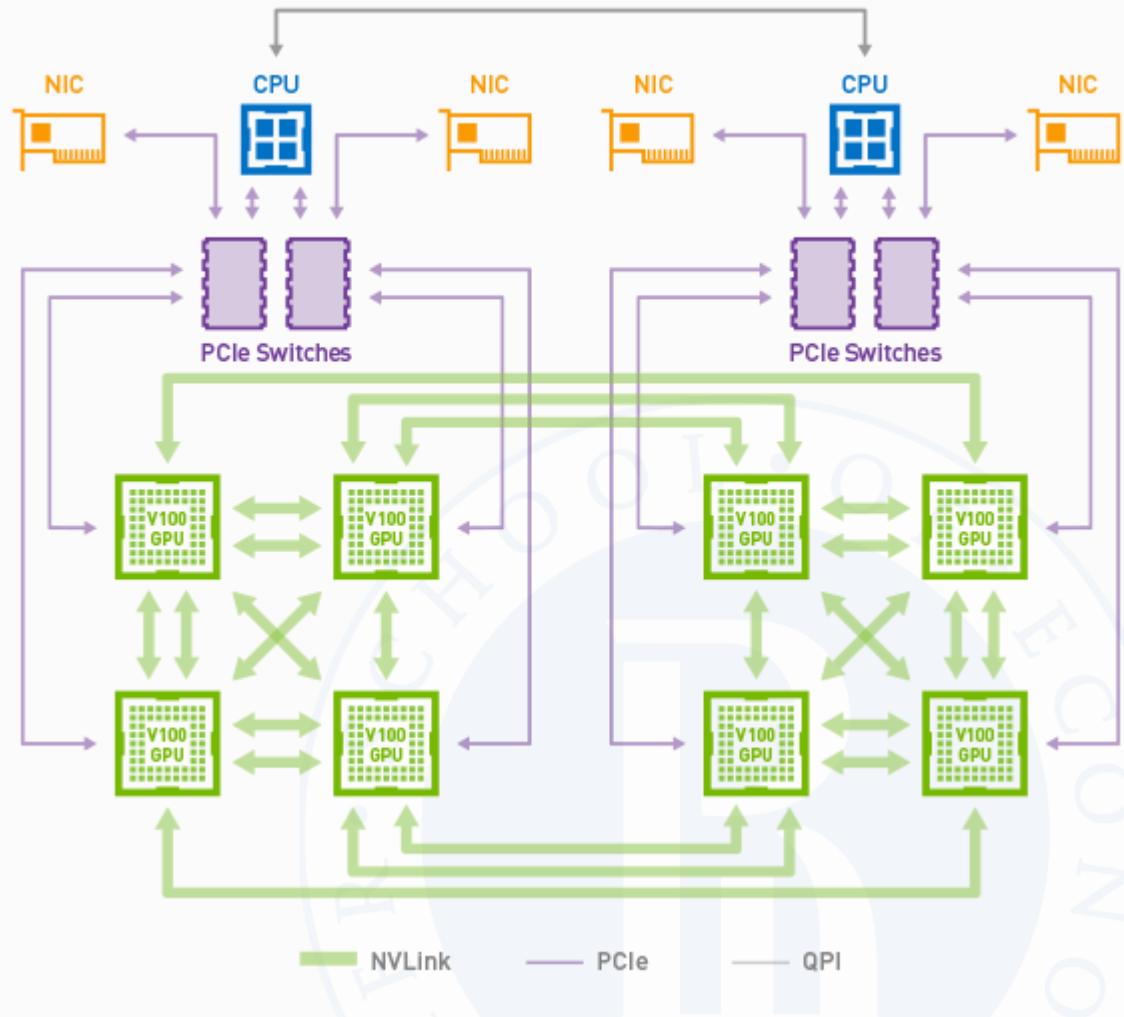
## → Простой случай

- Рассылаем копии модели на GPU
- Применяем модель на части батча
- Вклады в градиент отправляем координатору
- Суммируем и заново рассылаем



# Параллельная обработка данных

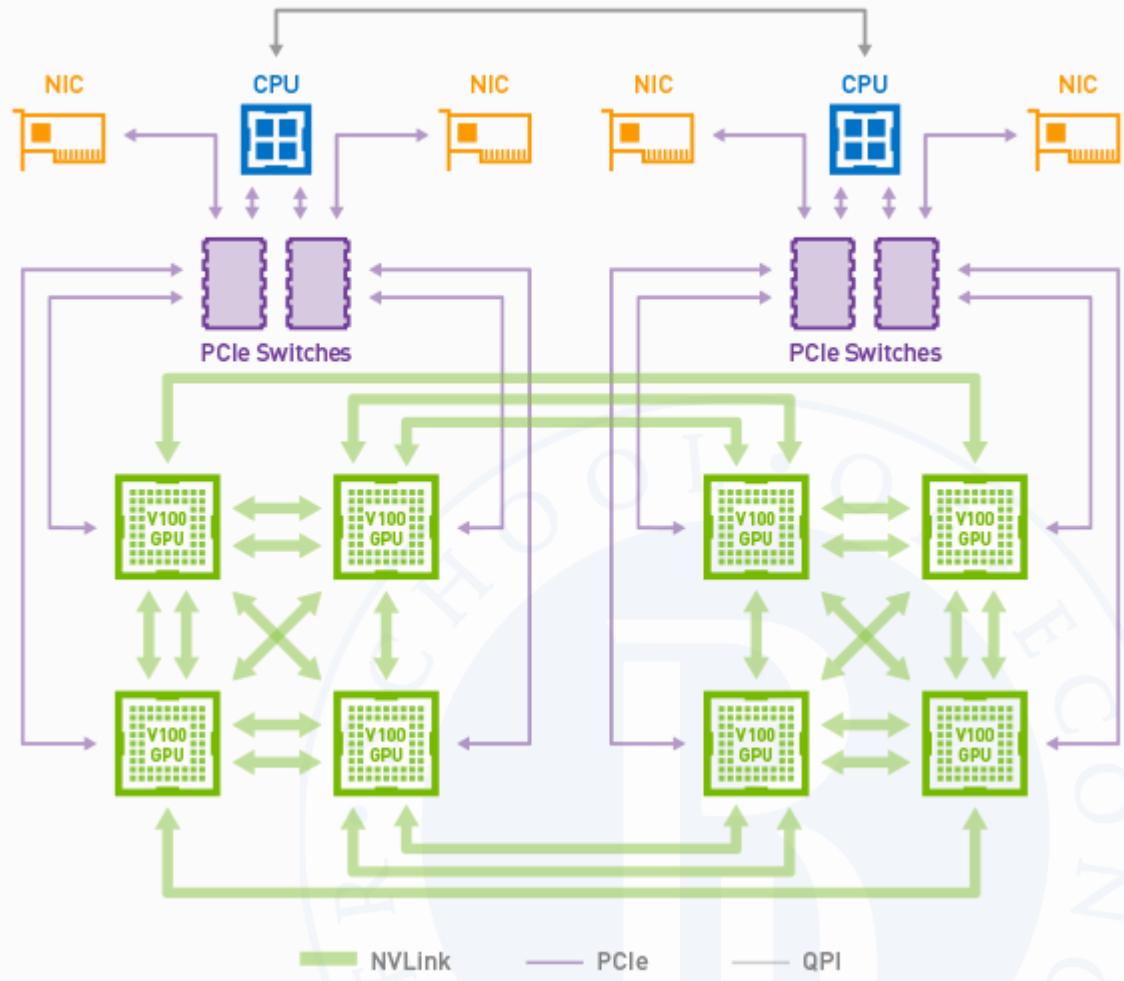
→ Сложный случай:



# Параллельная обработка данных

→ Сложный случай:

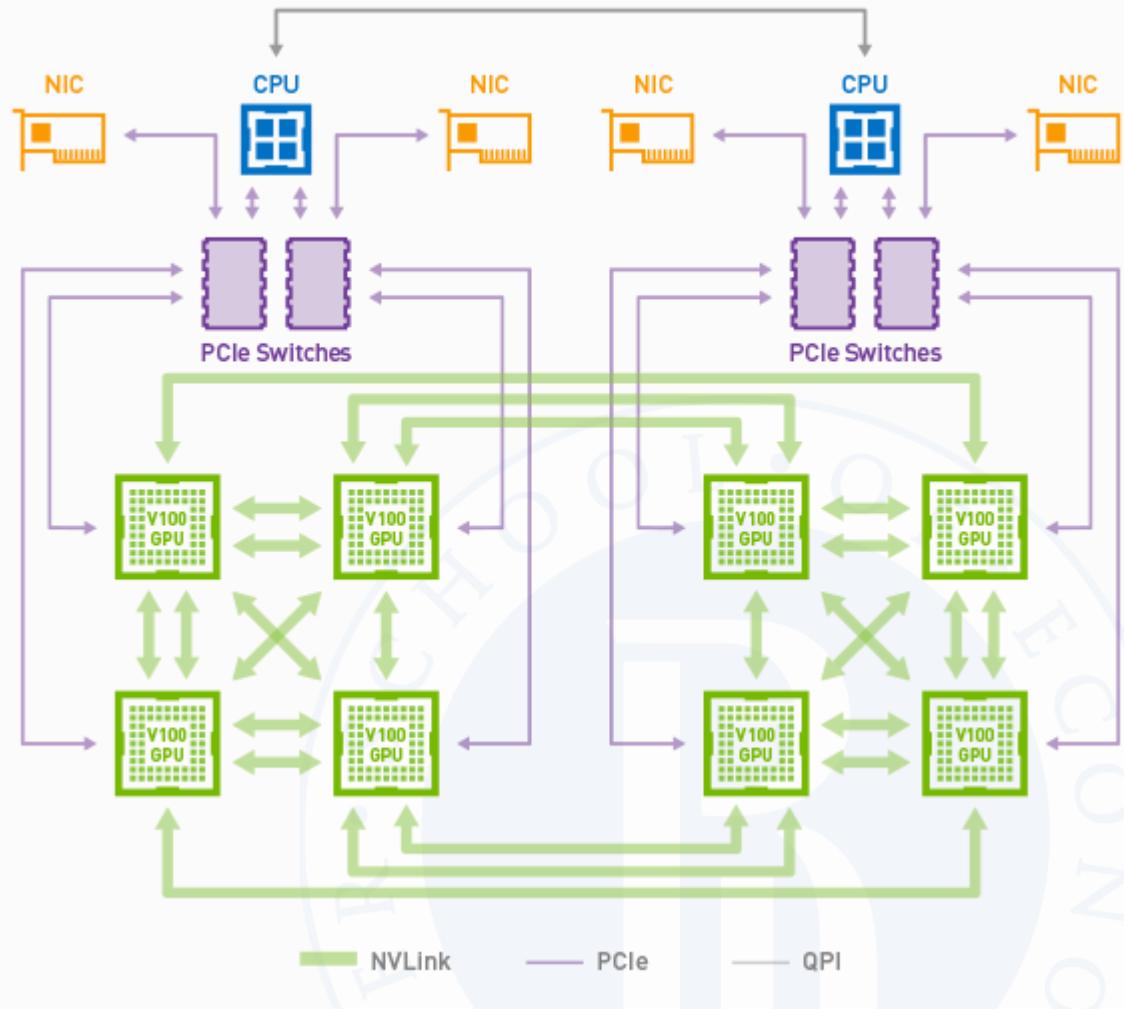
- кластер с одним устройством на каждом узле



# Параллельная обработка данных

→ Сложный случай:

- кластер с одним устройством на каждом узле
- кластер с несколькими устройствами на каждом узле



# Случай нескольких машин

→ В случае нескольких машин:

Координатор

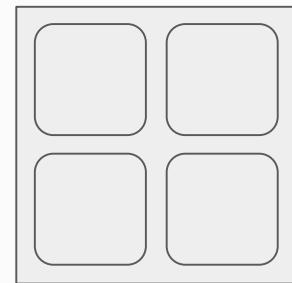
# Случай нескольких машин

→ В случае нескольких машин:

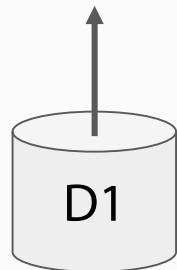
Координатор



Обработчики



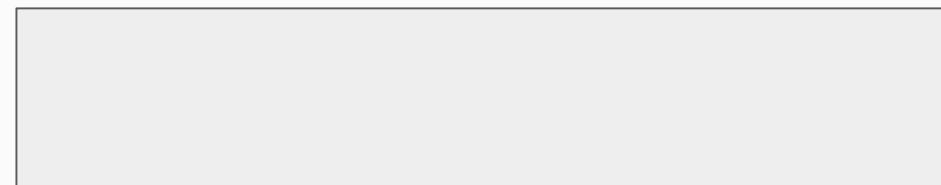
Данные



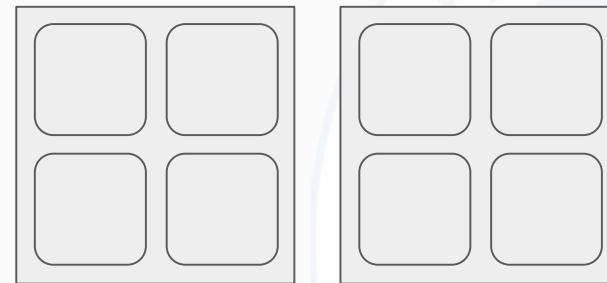
# Случай нескольких машин

→ В случае нескольких машин:

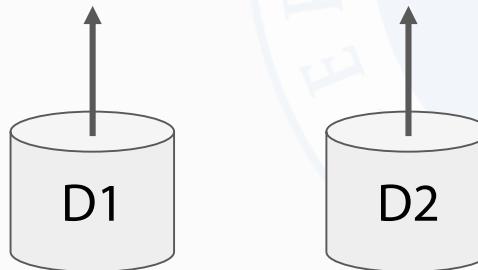
Координатор



Обработчики



Данные



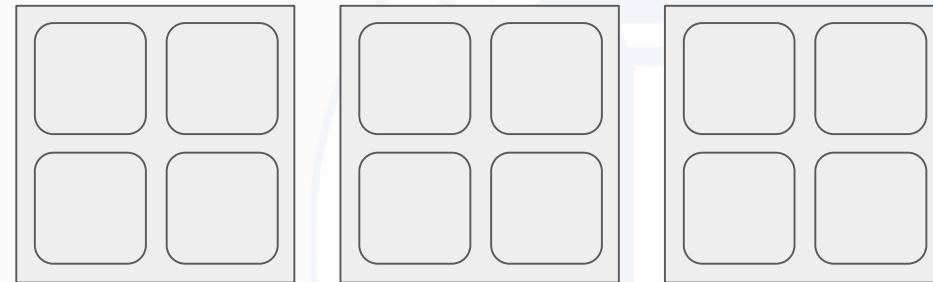
# Случай нескольких машин

→ В случае нескольких машин:

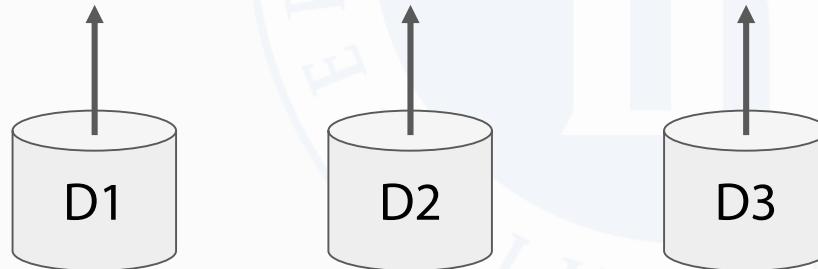
Координатор



Обработчики

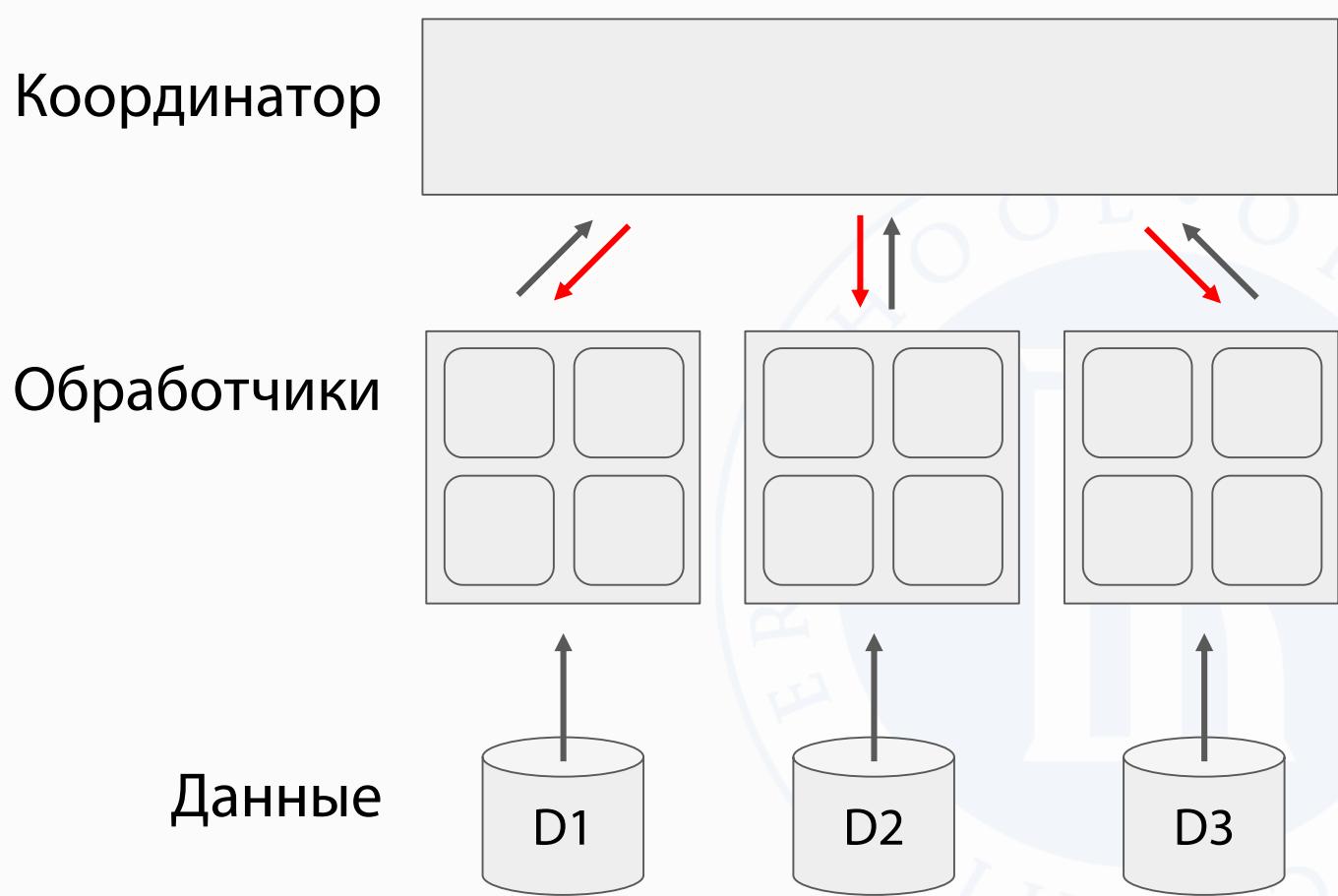


Данные



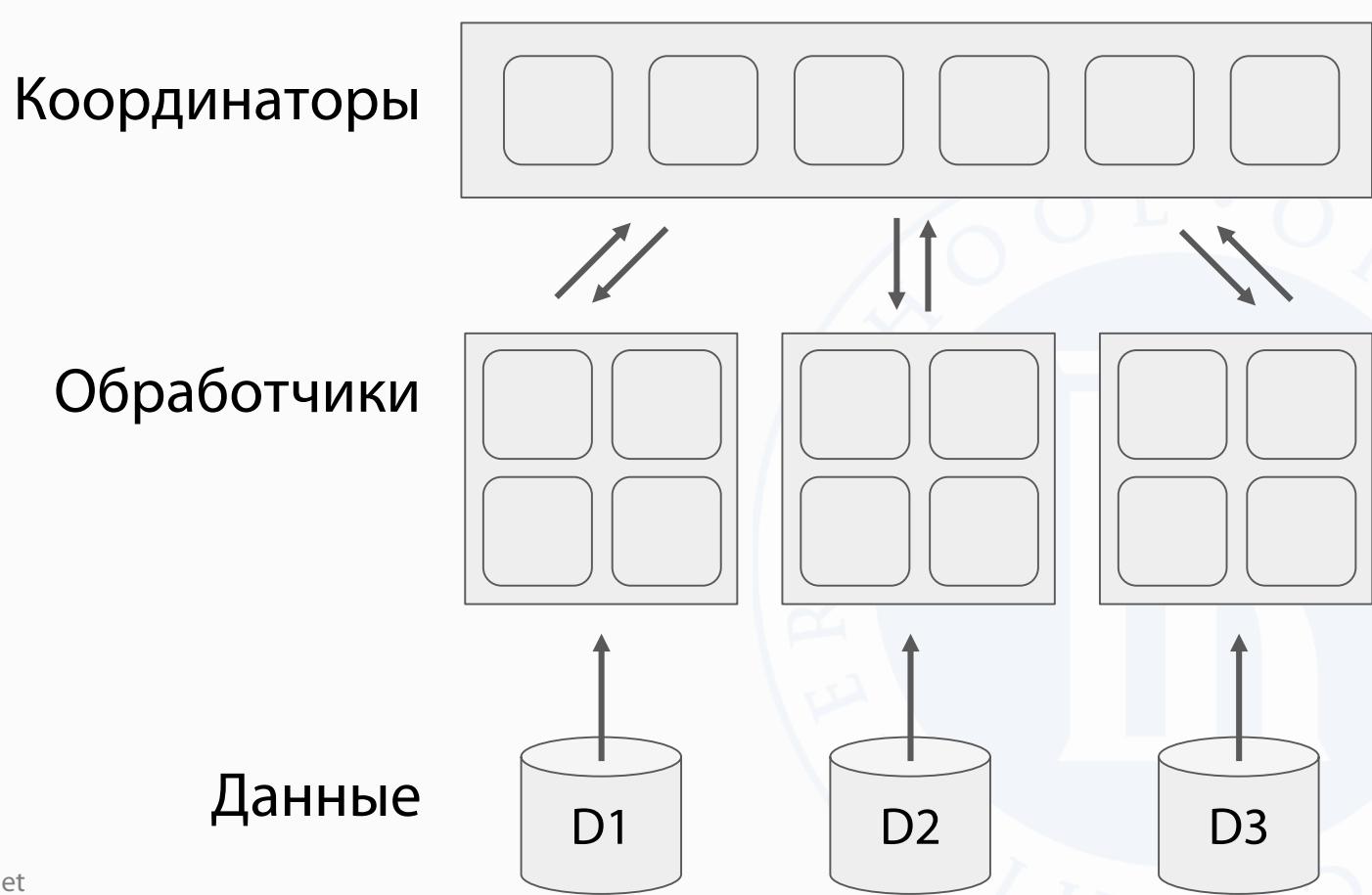
# Случай нескольких машин

- В случае нескольких машин:
- Обработчики запрашивают веса самостоятельно



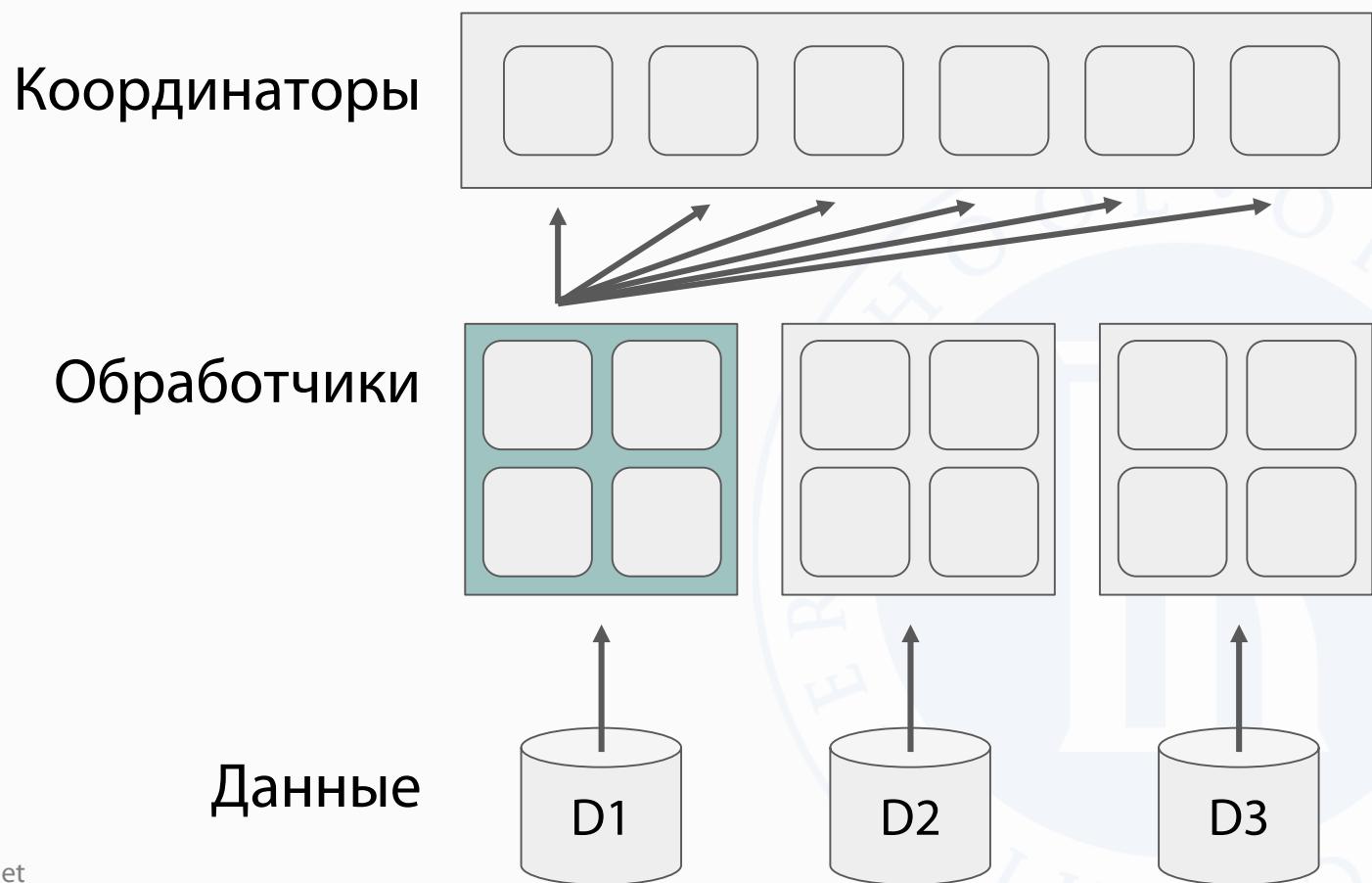
# Случай нескольких машин

→ Координаторов может быть несколько



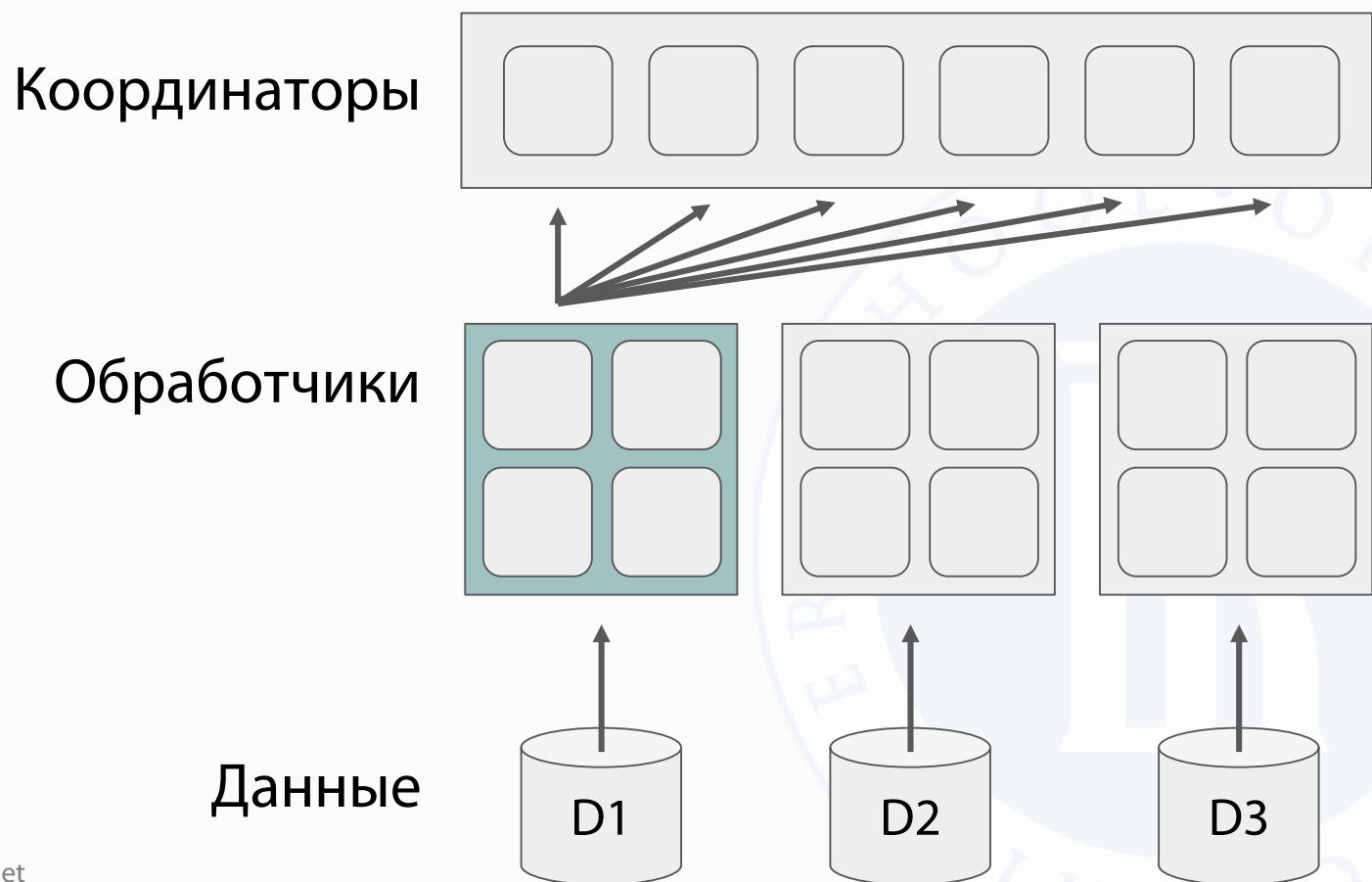
# Случай нескольких машин

- Координаторов может быть несколько
- Данные отправляются каждому координатору

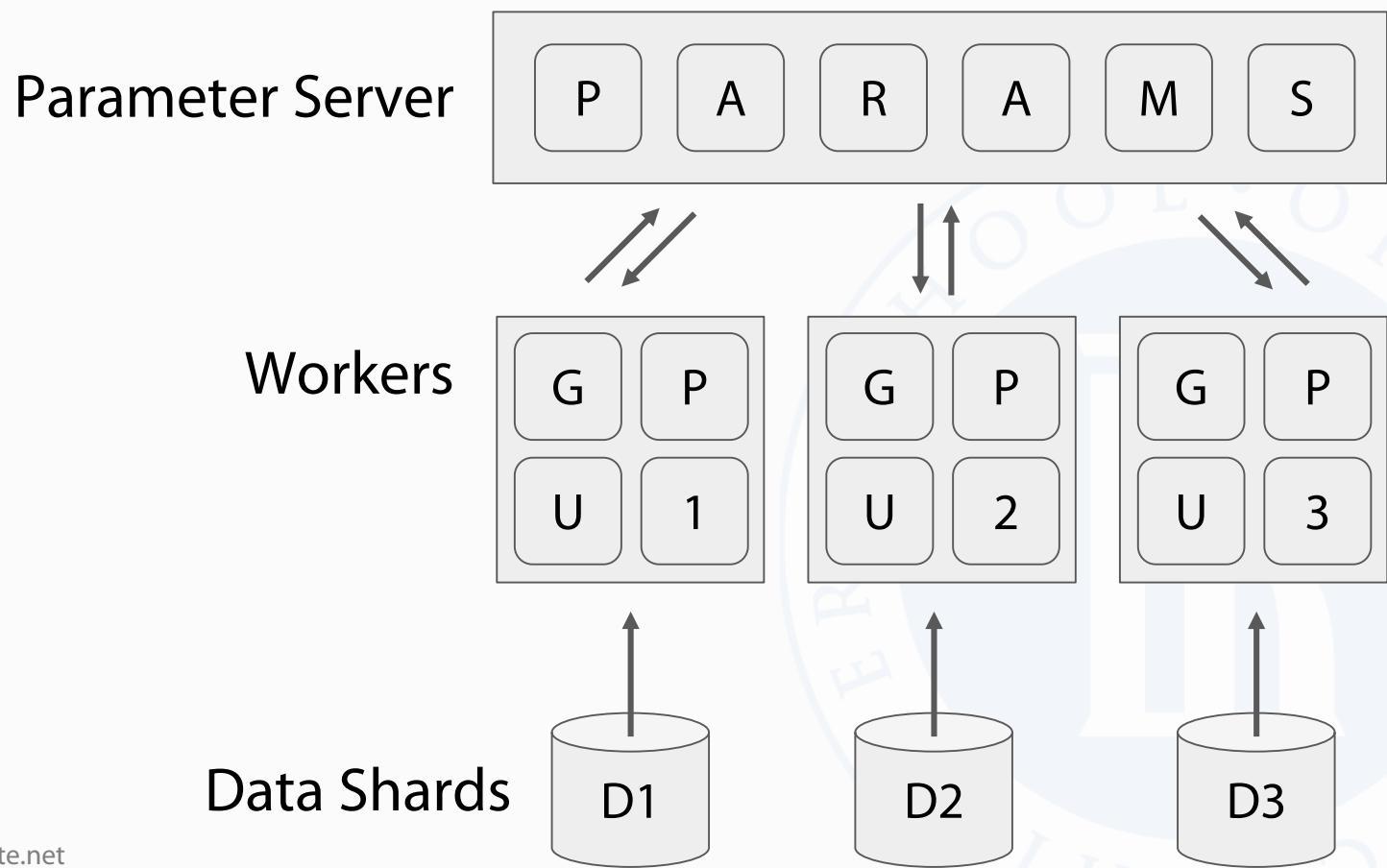


# Случай нескольких машин

- Координаторов может быть несколько
- Данные отправляются каждому координатору (allreduce)

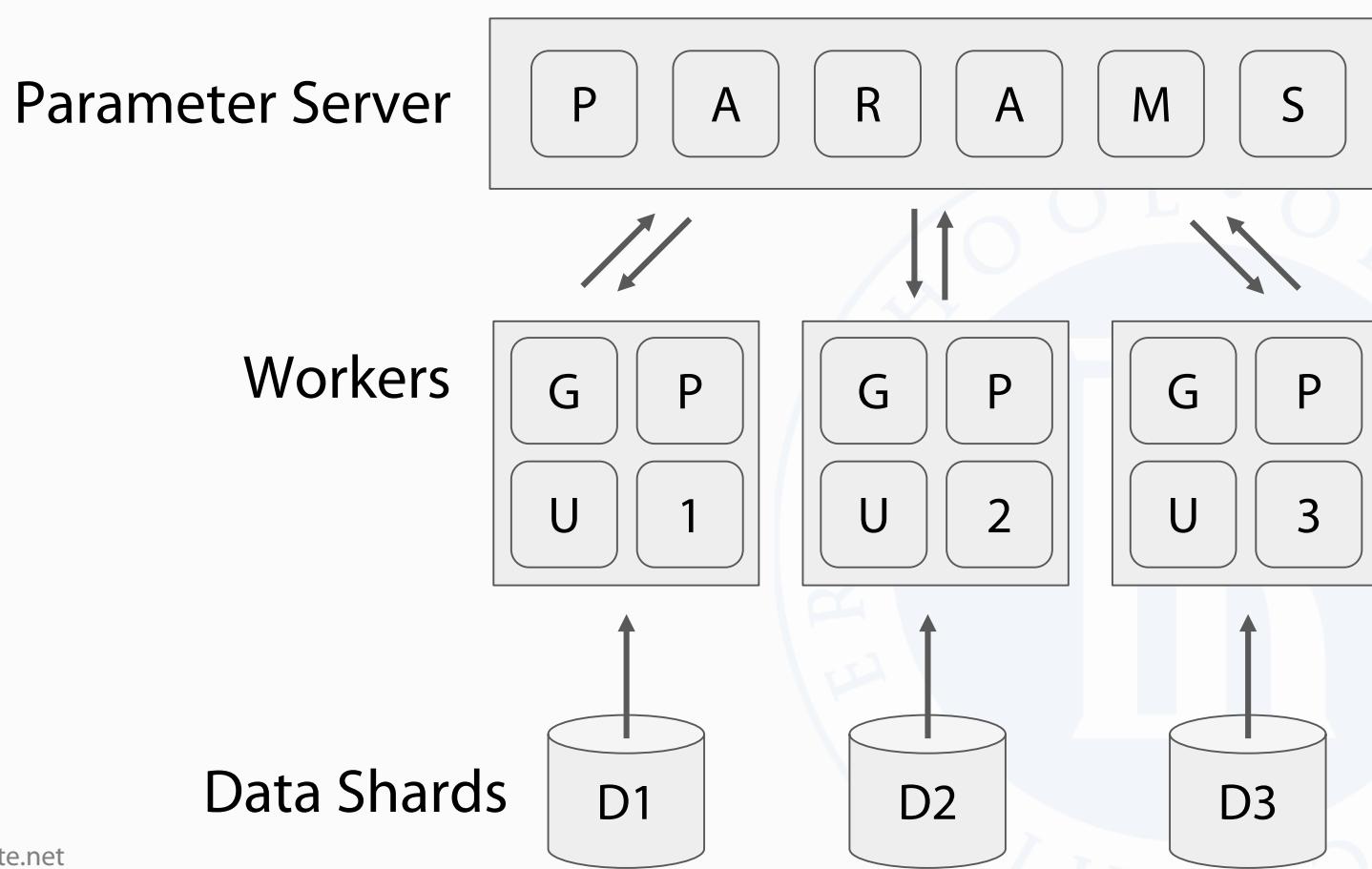


# Parameter Server



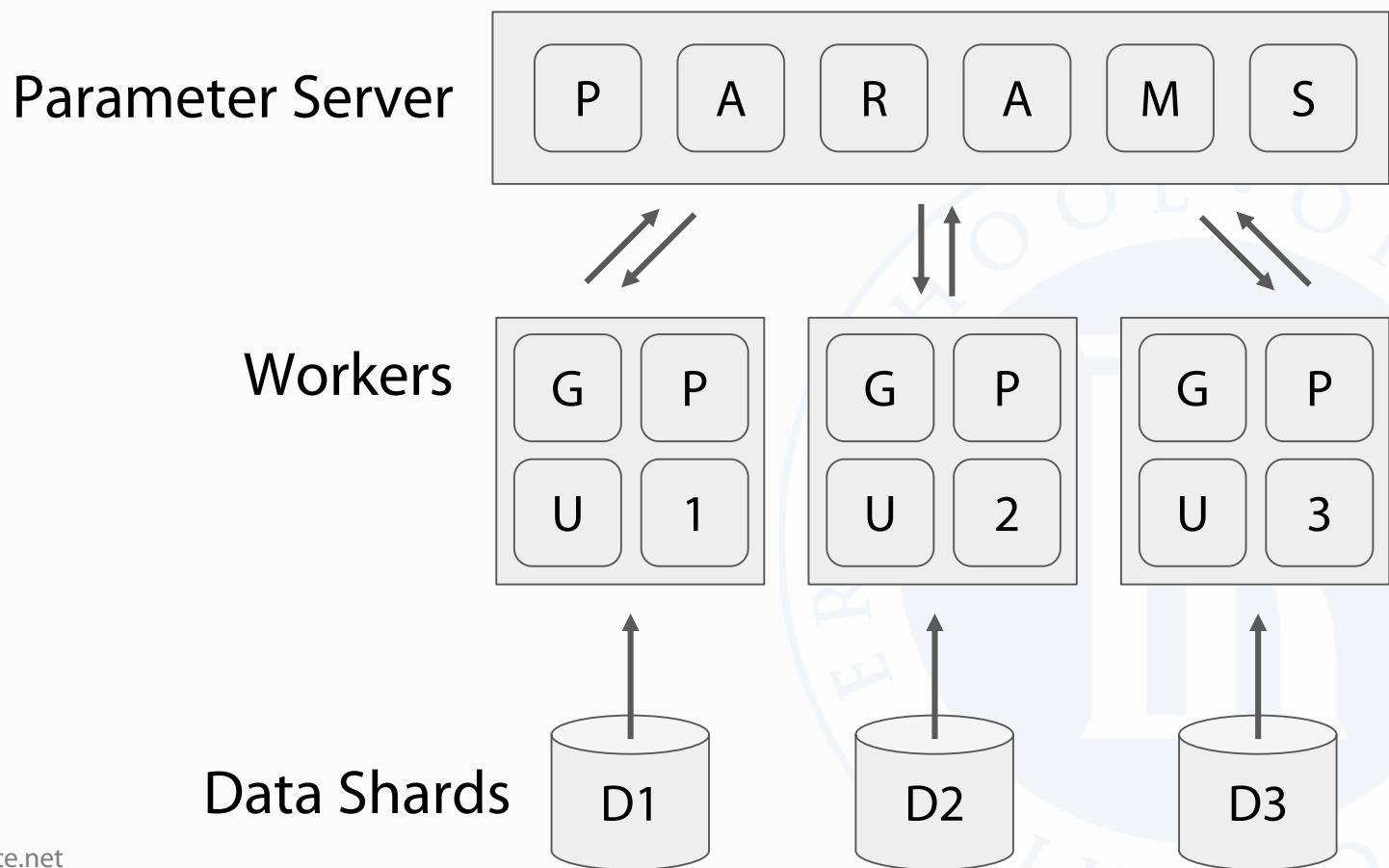
# Parameter Server

- Сервер параметров хранит веса



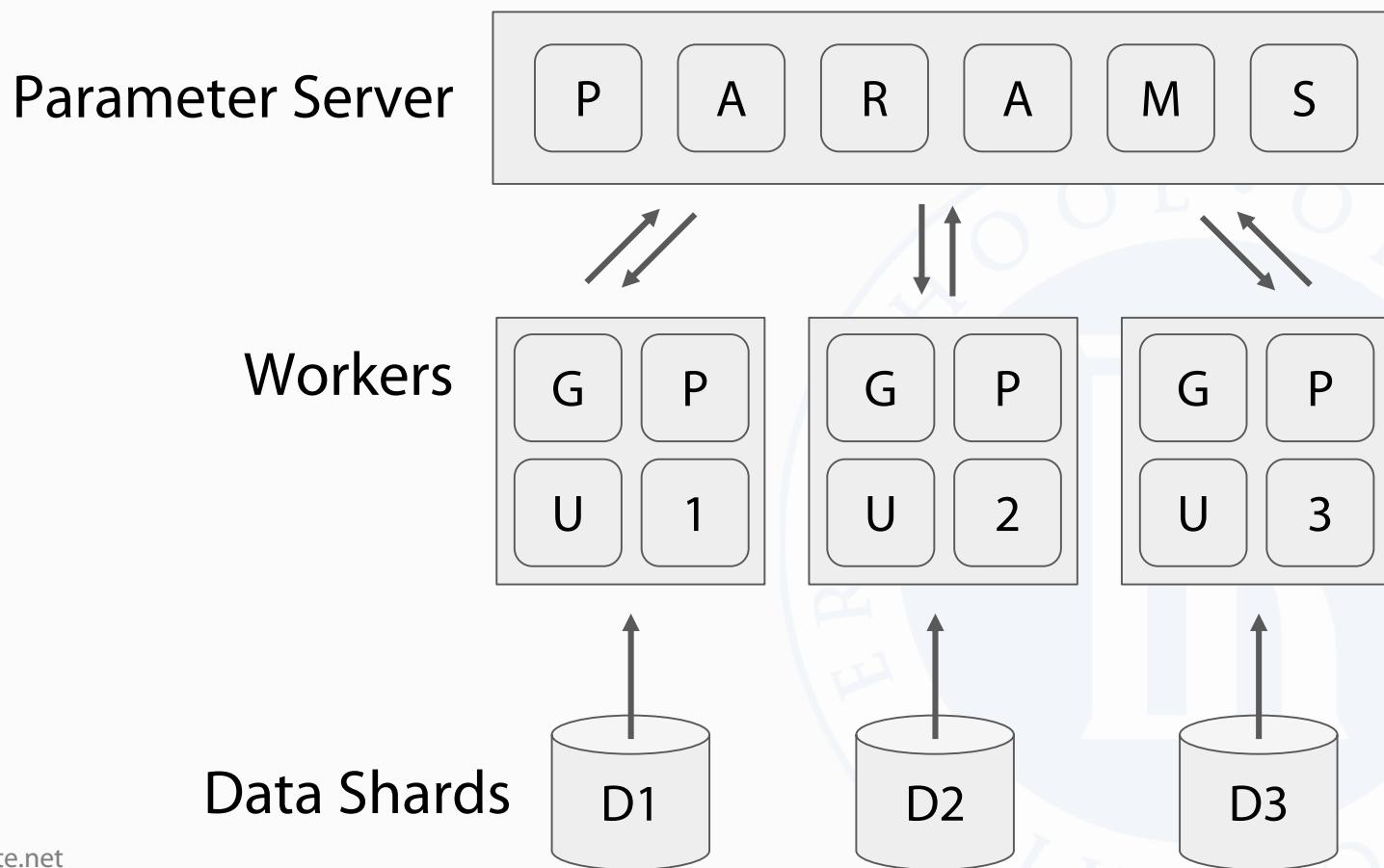
# Parameter Server

- Сервер параметров хранит веса
- Обработчики присылают изменения весов



# Parameter Server

- Сервер параметров хранит веса
- Обработчики присылают изменения весов
- Запрашивают новые веса



# Parameter Server

- Сервер параметров хранит веса
- Обработчики присылают изменения весов
- Запрашивают новые веса

Parameter Server



Workers

Какие проблемы?  
Как решать?

Data Shards

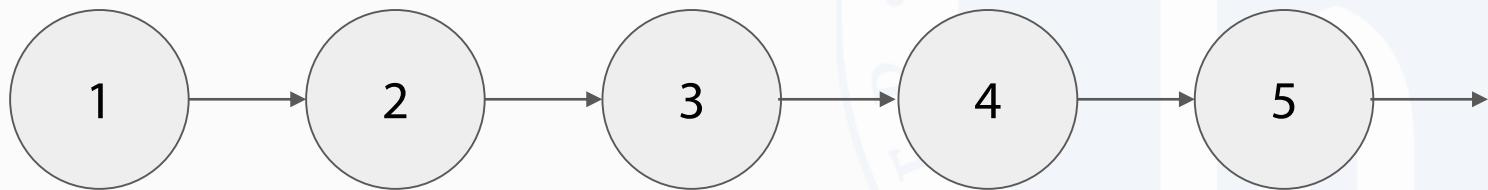
# PS. Консистентность и асинхронность



Упираемся в необходимость ожидать ответа от всех  
в координаторе



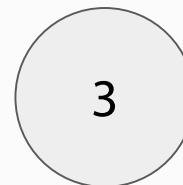
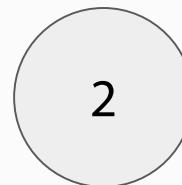
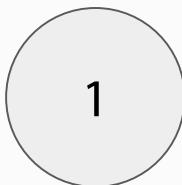
Каждая итерация ожидает ответа всех обработчиков  
и рассыпает собранный полностью ответ



# PS. Консистентность и асинхронность

→ Упираемся в необходимость ожидать ответа от всех в координаторе

→ Отменой ожидания можем регулировать **консистентность весов**



# PS. Консистентность и асинхронность

- Упираемся в необходимость ожидать ответа от всех в координаторе
- Отменой ожидания можем регулировать **консистентность весов**
- Делаем шаг с локальным весом **асинхронно**

1

2

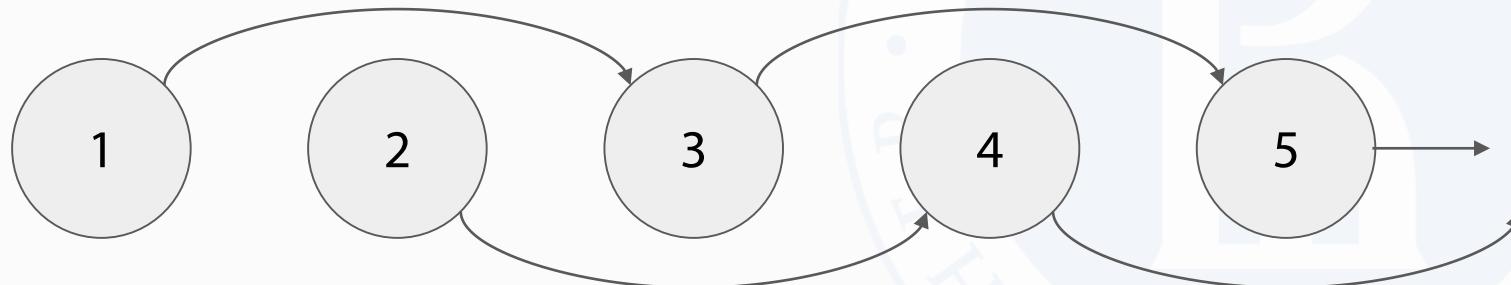
3

4

5

# PS. Консистентность и асинхронность

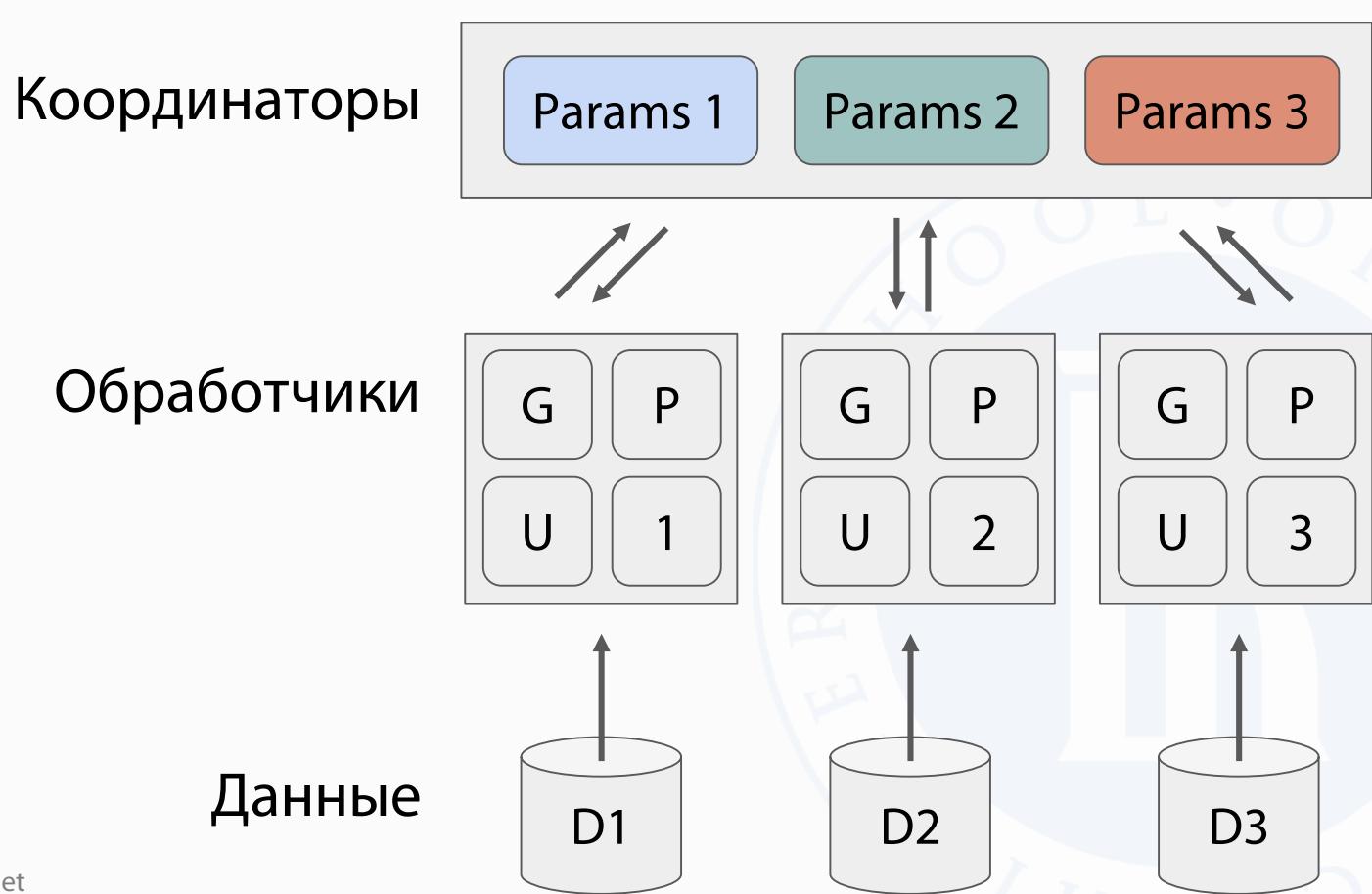
- Упираемся в необходимость ожидать ответа от всех в координаторе
- Отменой ожидания можем регулировать консистентность весов
- Делаем шаг с локальным весом асинхронно
- Можем ограничить отставание



Отставание 2

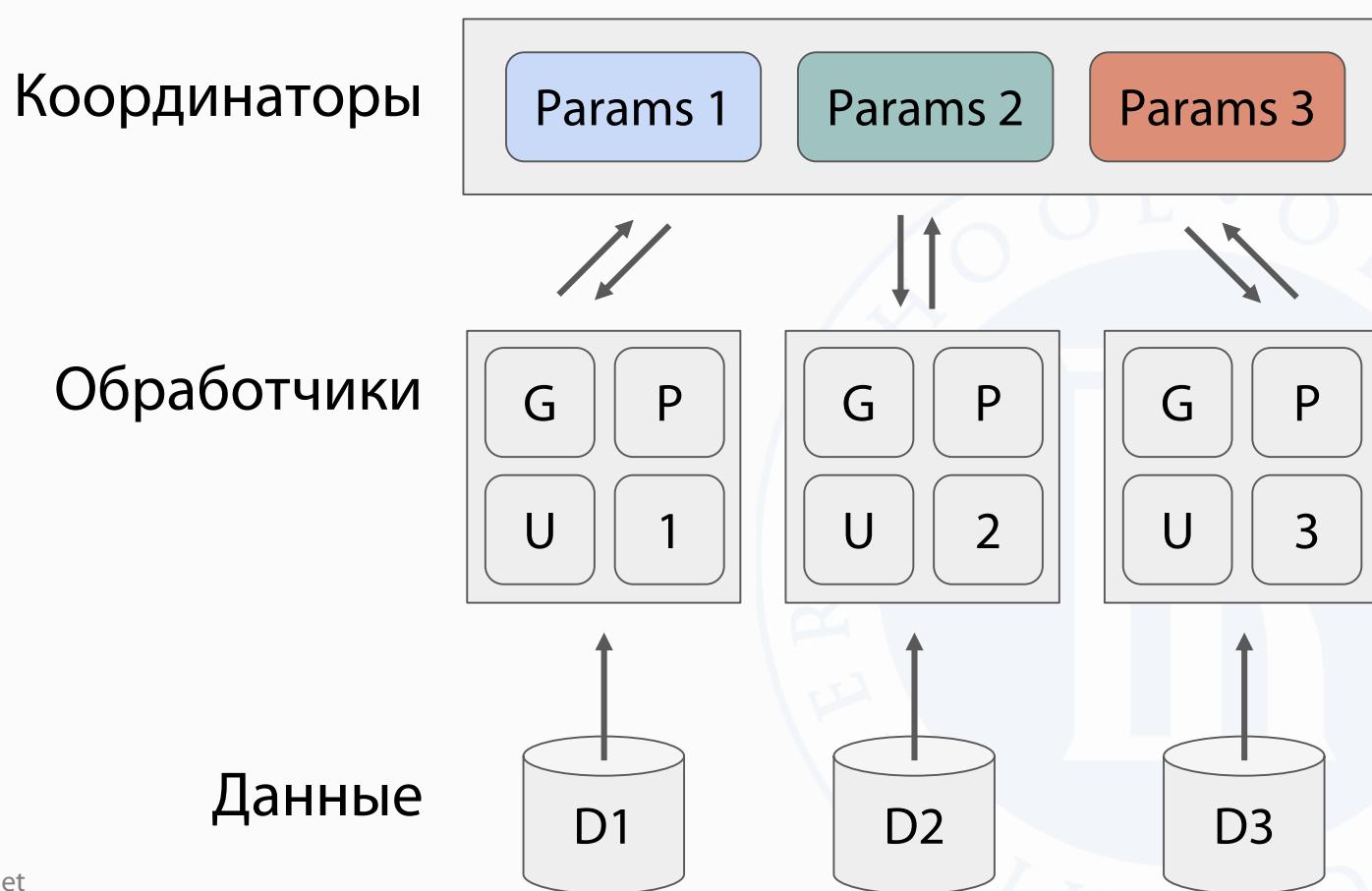
# PS. Шардирование параметров

- Много серверов — возможно шардирование параметров



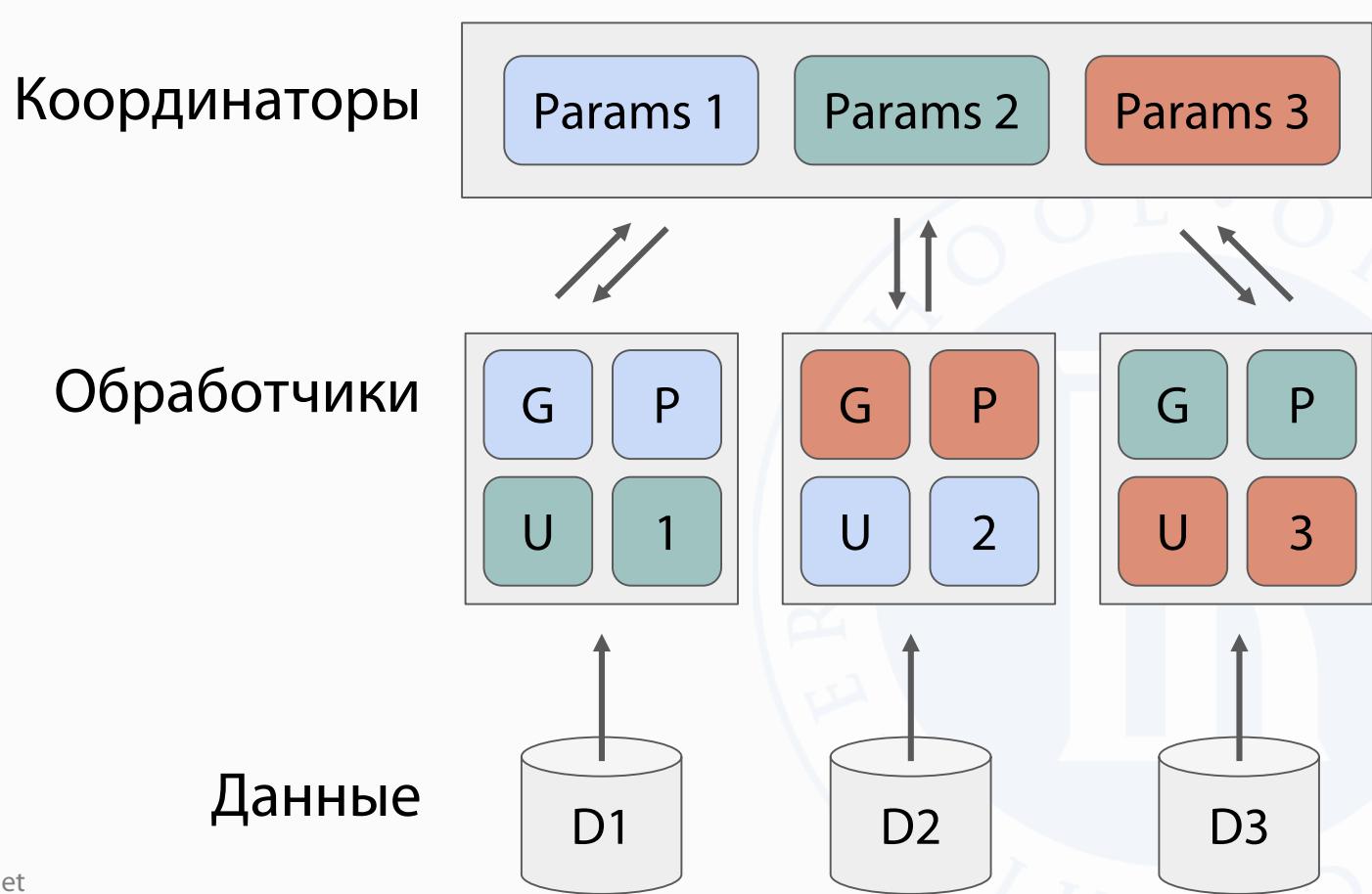
# PS. Шардирование параметров

- Много серверов — возможно шардирование параметров
- Каждый сервер отвечает за свою часть параметров



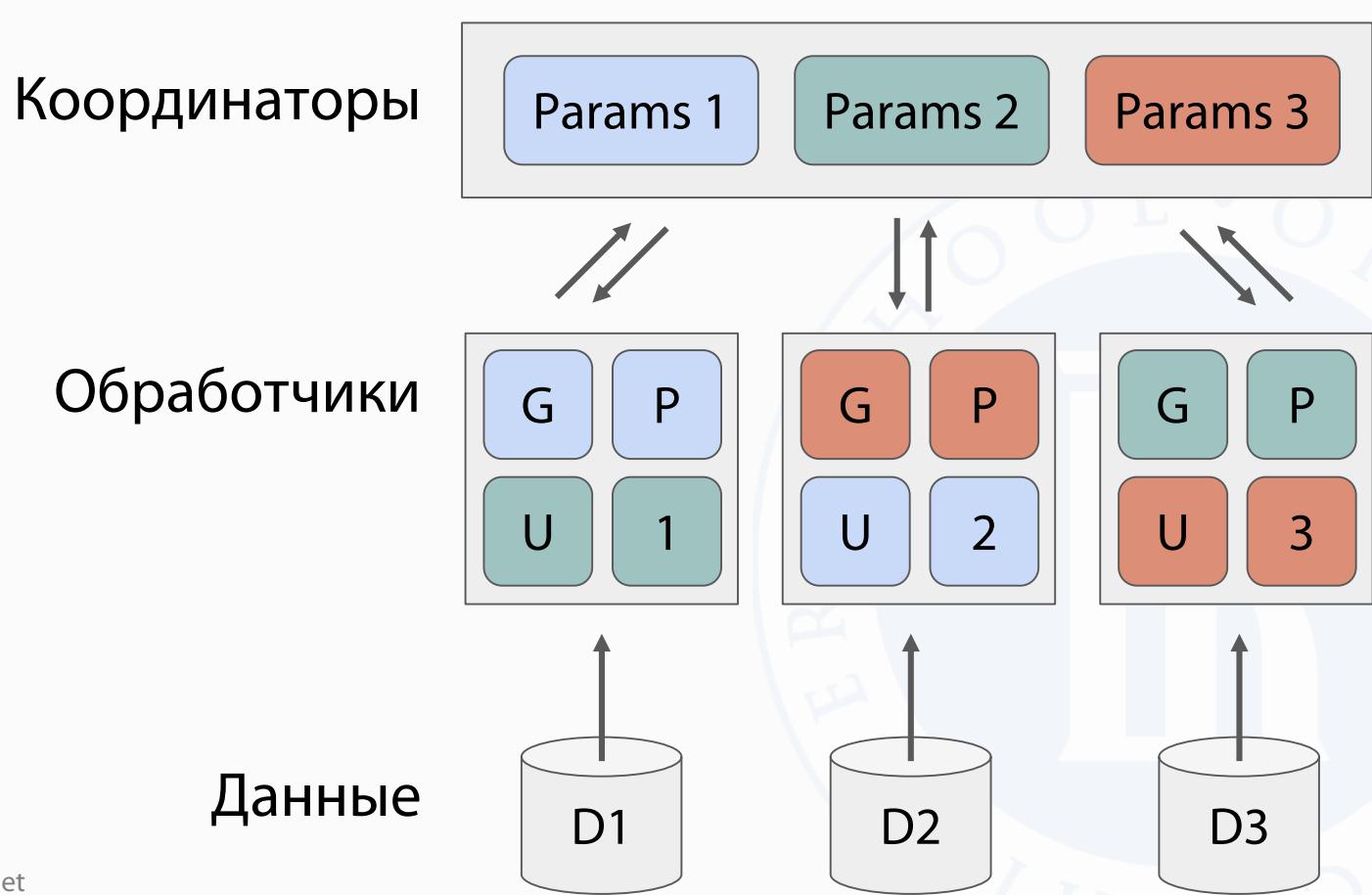
# PS. Асинхронный SGD

- Возможно получение части параметров



# PS. Асинхронный SGD

- Возможно получение части параметров
- Экономит время и число взаимодействий с серверами



# PS. Distributed Delayed Proximal Gradient

→ Пусть у нас есть набор параметров, разделим на  $B$  блоков  
Установим порядок  $b(1), \dots, b(T)$  и максимальную  
задержку  $\tau$

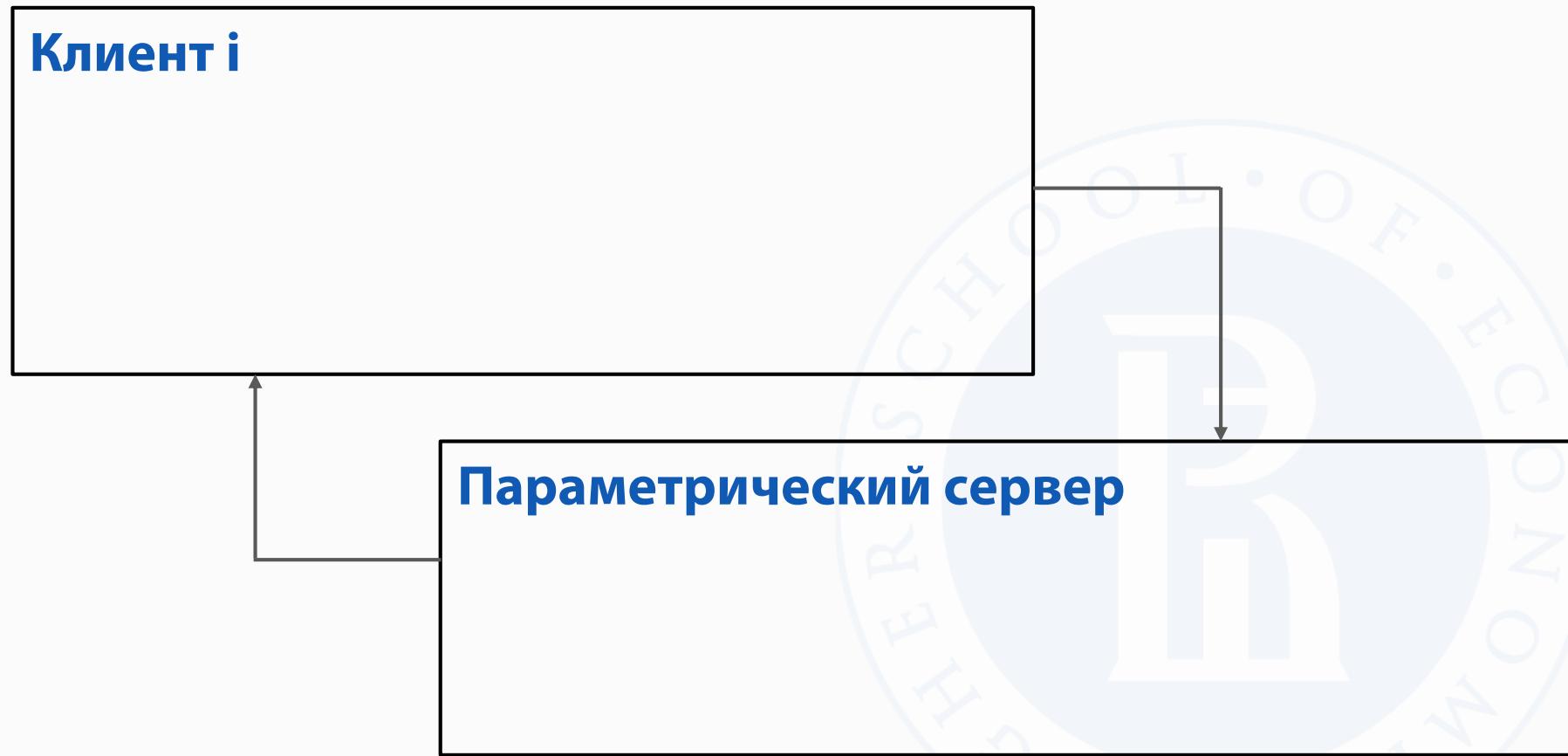
# PS. Distributed Delayed Proximal Gradient

→ Пусть у нас есть набор параметров, разделим на  $B$  блоков  
Установим порядок  $b(1), \dots, b(T)$  и максимальную  
задержку  $\tau$

Клиент  $i$

# PS. Distributed Delayed Proximal Gradient

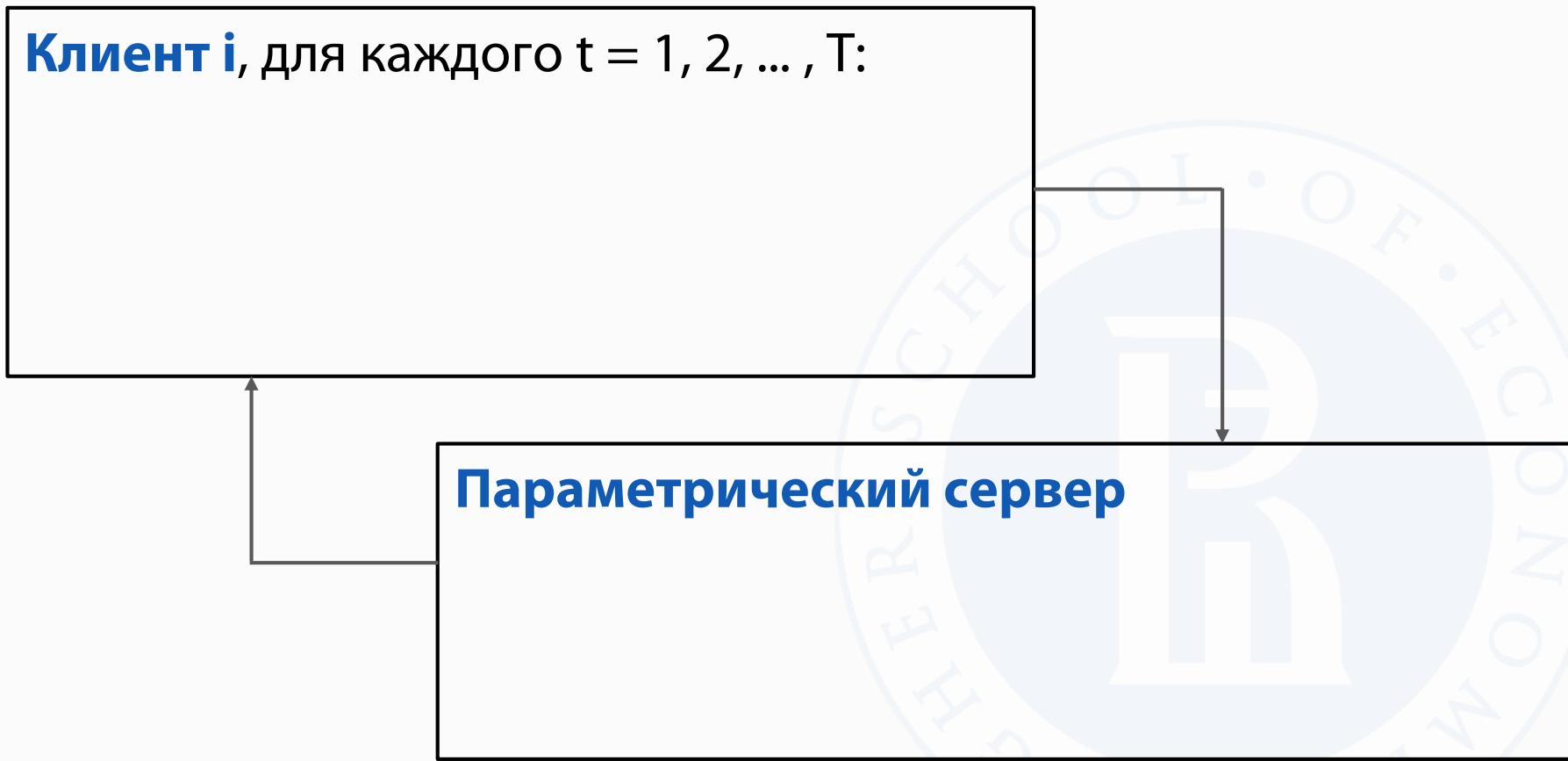
→ Пусть у нас есть набор параметров, разделим на  $B$  блоков  
Установим порядок  $b(1), \dots, b(T)$  и максимальную  
задержку  $\tau$



# PS. Distributed Delayed Proximal Gradient

→ Пусть у нас есть набор параметров, разделим на  $B$  блоков  
Установим порядок  $b(1), \dots, b(T)$  и максимальную  
задержку  $\tau$

**Клиент  $i$** , для каждого  $t = 1, 2, \dots, T$ :



# PS. Distributed Delayed Proximal Gradient

→ Пусть у нас есть набор параметров, разделим на  $B$  блоков  
Установим порядок  $b(1), \dots, b(T)$  и максимальную  
задержку  $\tau$

**Клиент  $i$** , для каждого  $t = 1, 2, \dots, T$ :

- расчет градиента

**Параметрический сервер**



# PS. Distributed Delayed Proximal Gradient

→ Пусть у нас есть набор параметров, разделим на  $B$  блоков  
Установим порядок  $b(1), \dots, b(T)$  и максимальную  
задержку  $\tau$

**Клиент  $i$** , для каждого  $t = 1, 2, \dots, T$ :

- расчет градиента
- отправка градиента  
параметрическому серверу

**Параметрический сервер**

# PS. Distributed Delayed Proximal Gradient

→ Пусть у нас есть набор параметров, разделим на  $B$  блоков  
Установим порядок  $b(1), \dots, b(T)$  и максимальную  
задержку  $\tau$

**Клиент  $i$** , для каждого  $t = 1, 2, \dots, T$ :

- расчет градиента
- отправка градиента  
параметрическому серверу
- ожидание получения  $x(t - \tau)$

**Параметрический сервер**

# PS. Distributed Delayed Proximal Gradient

→ Пусть у нас есть набор параметров, разделим на  $B$  блоков  
Установим порядок  $b(1), \dots, b(T)$  и максимальную  
задержку  $\tau$

**Клиент  $i$** , для каждого  $t = 1, 2, \dots, T$ :

- расчет градиента
- отправка градиента  
параметрическому серверу
- ожидание получения  $x(t - \tau)$

**Параметрический сервер**, для  $t = 1, 2, \dots, T$ :



# PS. Distributed Delayed Proximal Gradient

→ Пусть у нас есть набор параметров, разделим на  $B$  блоков  
Установим порядок  $b(1), \dots, b(T)$  и максимальную  
задержку  $\tau$

**Клиент  $i$** , для каждого  $t = 1, 2, \dots, T$ :

- расчет градиента
- отправка градиента  
параметрическому серверу
- ожидание получения  $x(t - \tau)$

**Параметрический сервер**, для  $t = 1, 2, \dots, T$ :

- принять приблизительный градиент

# PS. Distributed Delayed Proximal Gradient

→ Пусть у нас есть набор параметров, разделим на  $B$  блоков  
Установим порядок  $b(1), \dots, b(T)$  и максимальную  
задержку  $\tau$

**Клиент  $i$** , для каждого  $t = 1, 2, \dots, T$ :

- расчет градиента
- отправка градиента  
параметрическому серверу
- ожидание получения  $x(t - \tau)$

**Параметрический сервер**, для  $t = 1, 2, \dots, T$ :

- принять приблизительный градиент
- обновить  $x$  в блоке  $b(t)$

# PS. Distributed Delayed Proximal Gradient

→ Пусть у нас есть набор параметров, разделим на  $B$  блоков  
Установим порядок  $b(1), \dots, b(T)$  и максимальную  
задержку  $\tau$

**Клиент  $i$** , для каждого  $t = 1, 2, \dots, T$ :

- расчет градиента
- отправка градиента  
параметрическому серверу
- ожидание получения  $x(t - \tau)$

**Параметрический сервер**, для  $t = 1, 2, \dots, T$ :

- принять приблизительный градиент
- обновить  $x$  в блоке  $b(t)$
- отправить обновления клиентам

# PS. Distributed Delayed Proximal Gradient

- Пример работы:
  - На каждой  $t$ -ой итерации обновляем веса в блоке  $b(t)$

# PS. Distributed Delayed Proximal Gradient

- Пример работы:
  - На каждой  $t$ -ой итерации обновляем веса в блоке  $b(t)$
  - Хотим отставать не более, чем на  $\tau$  блоков

# PS. Distributed Delayed Proximal Gradient



Пример работы:

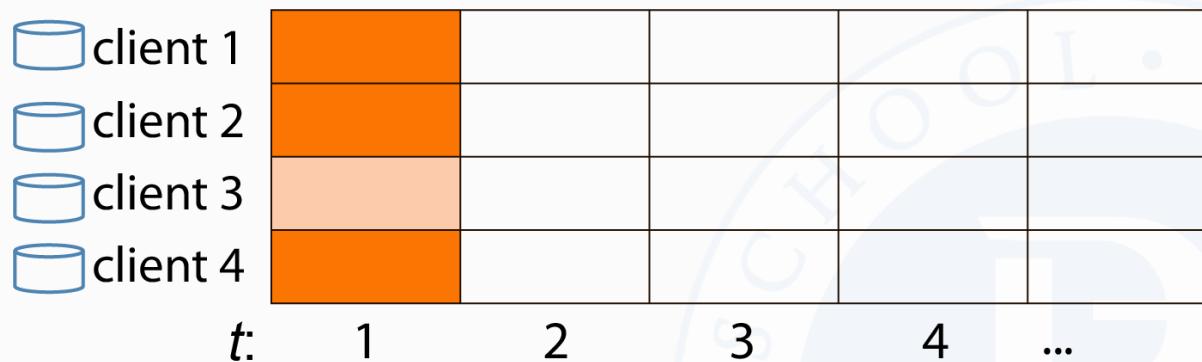
- На каждой  $t$ -ой итерации обновляем веса в блоке  $b(t)$
- Хотим отставать не более, чем на  $\tau$  блоков
- Обновления весов посылаются асинхронно

# PS. Distributed Delayed Proximal Gradient

- Пример работы при  $\tau = 2$ :
  - На каждой  $t$ -ой итерации обновляем веса в блоке  $b(t)$
  - Хотим отставать не более, чем на  $\tau$  блоков
  - Обновления весов посылаются асинхронно

# PS. Distributed Delayed Proximal Gradient

- Пример работы при  $\tau = 2$ :
- На каждой  $t$ -ой итерации обновляем веса в блоке  $b(t)$
  - Хотим отставать не более, чем на  $\tau$  блоков
  - Обновления весов посылаются асинхронно



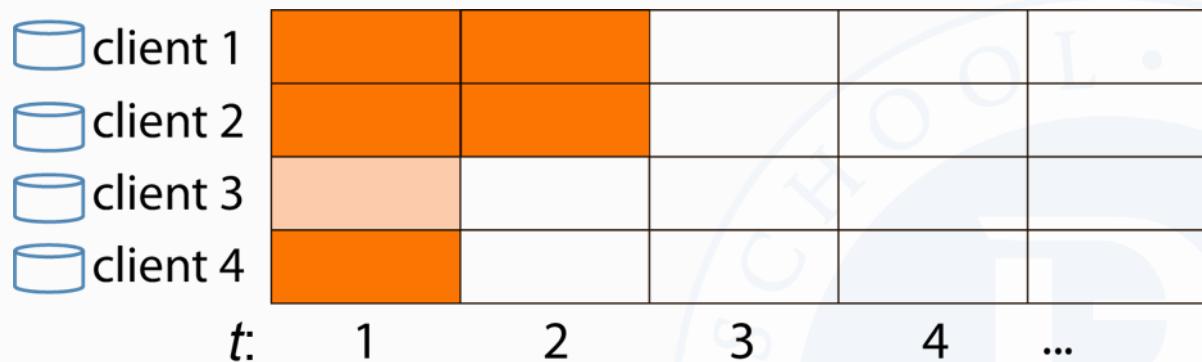
Предположим, что к какому-то моменту первую итерацию завершили клиенты 1, 2 и 4,

# PS. Distributed Delayed Proximal Gradient



Пример работы при  $\tau = 2$ :

- На каждой  $t$ -ой итерации обновляем веса в блоке  $b(t)$
- Хотим отставать не более, чем на  $\tau$  блоков
- Обновления весов посылаются асинхронно



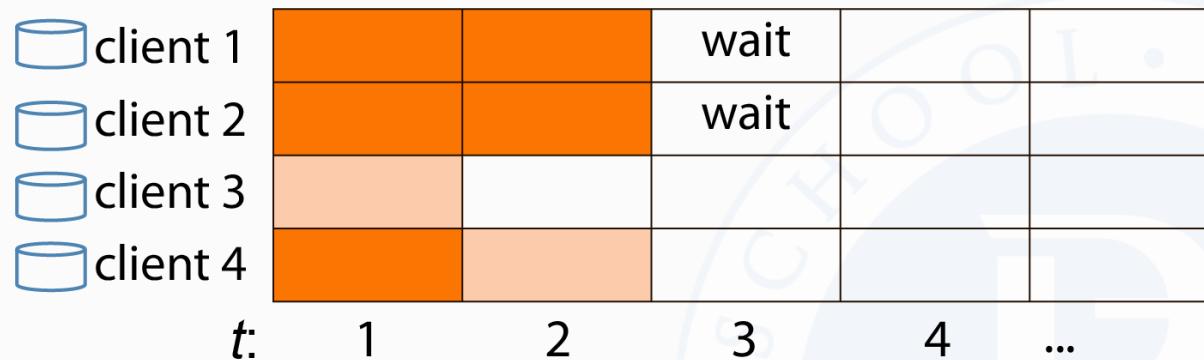
Предположим, что к какому-то моменту первую итерацию завершили клиенты 1, 2 и 4, затем клиенты 1 и 2 завершили итерацию 2

# PS. Distributed Delayed Proximal Gradient



Пример работы при  $\tau = 2$ :

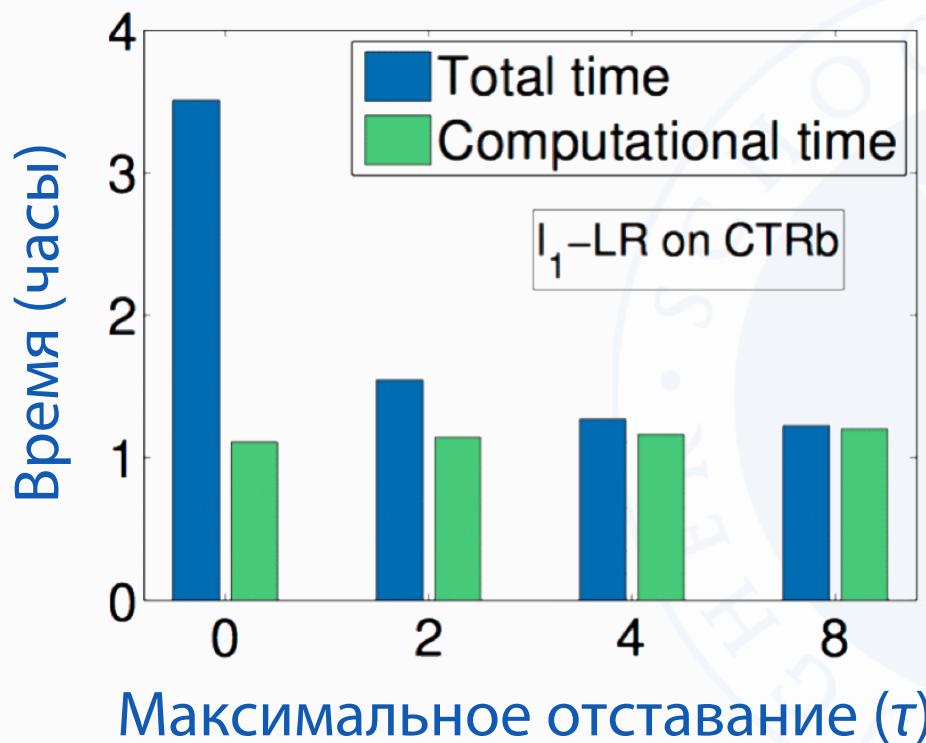
- На каждой  $t$ -ой итерации обновляем веса в блоке  $b(t)$
- Хотим отставать не более, чем на  $\tau$  блоков
- Обновления весов посылаются асинхронно



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

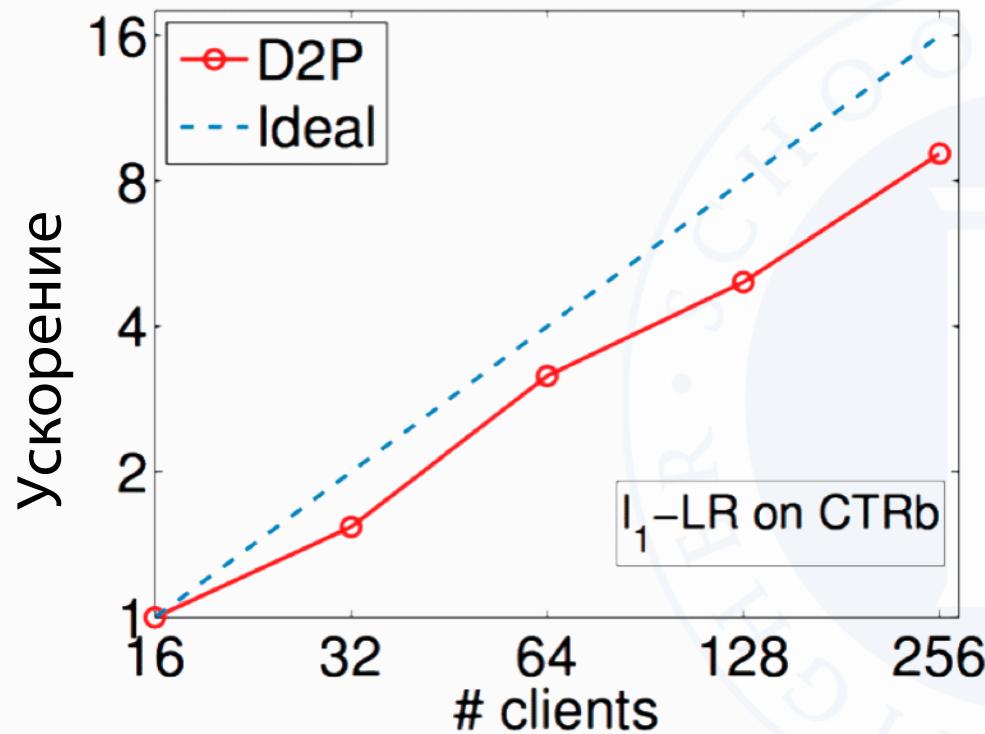
# PS. Distributed Delayed Proximal Gradient

- Получаем в  $\tau$  раз меньше синхронизаций
- Замедление сходимости компенсируется ускорением одного шага



# PS. Distributed Delayed Proximal Gradient

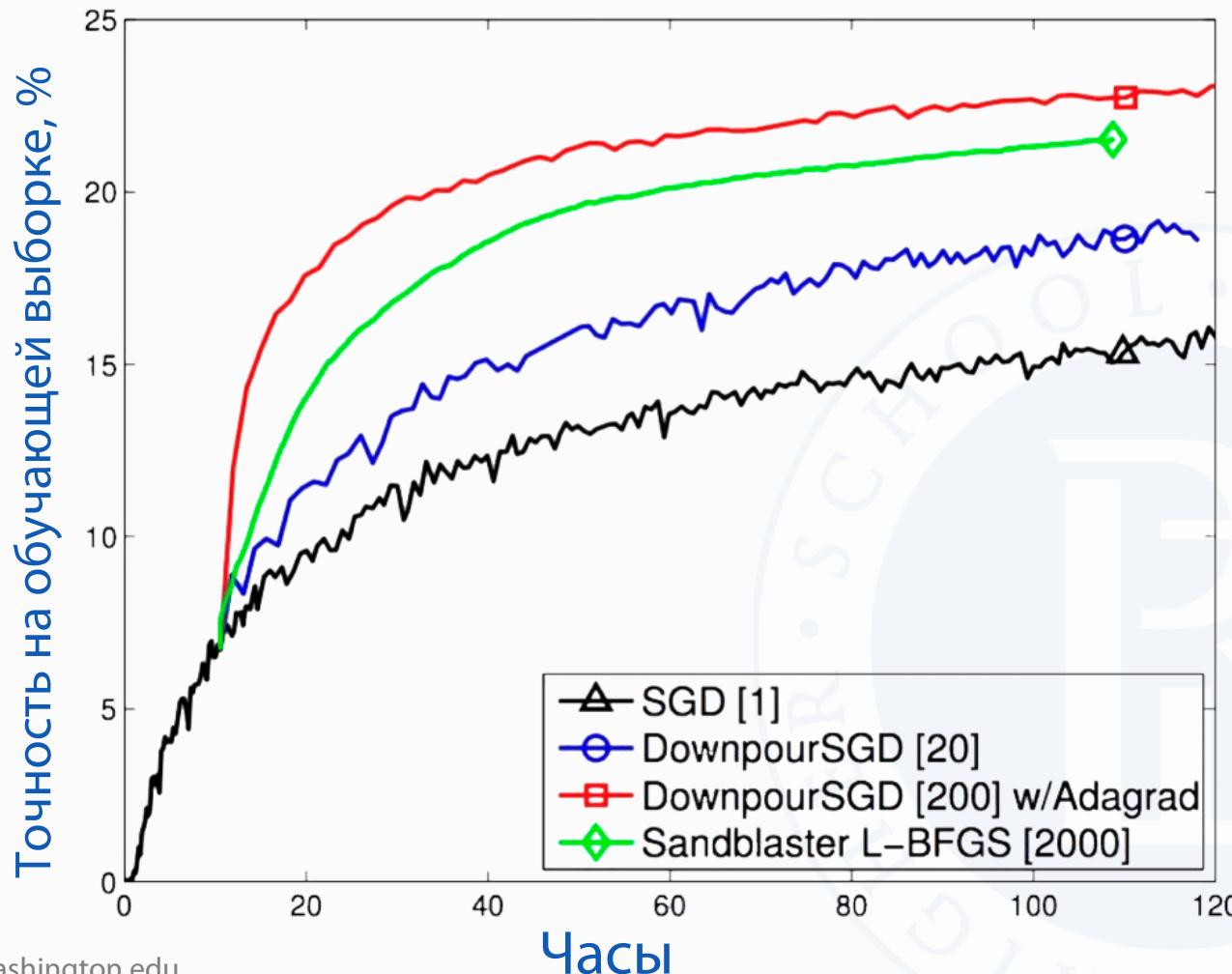
- Получаем в  $\tau$  раз меньше синхронизаций
- Замедление сходимости компенсируется ускорением одного шага



# Асинхронный SGD от Google (DistBelief)

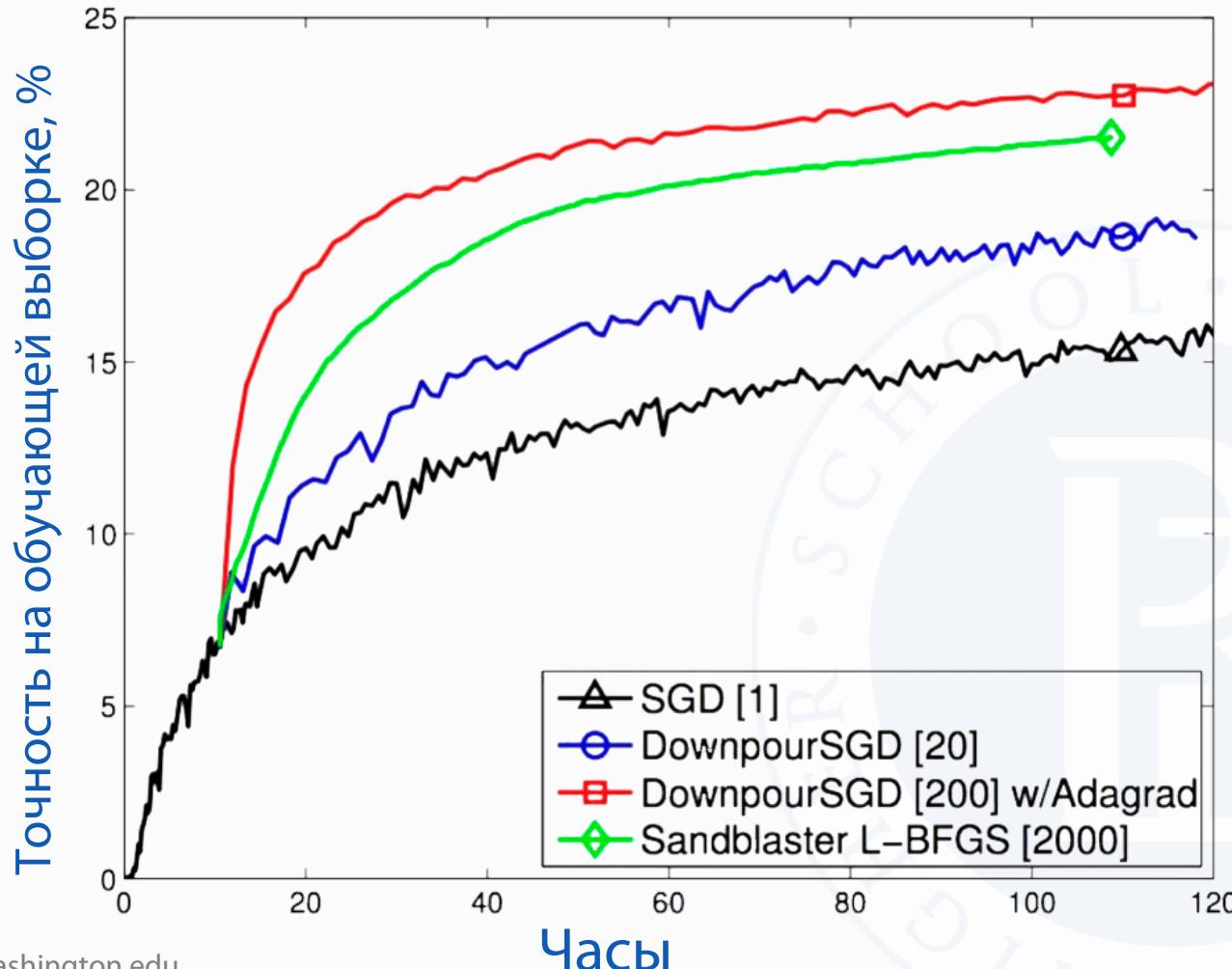


Увеличиваем размер блока данных, обучаем асинхронно



# Асинхронный SGD от Google (DistBelief)

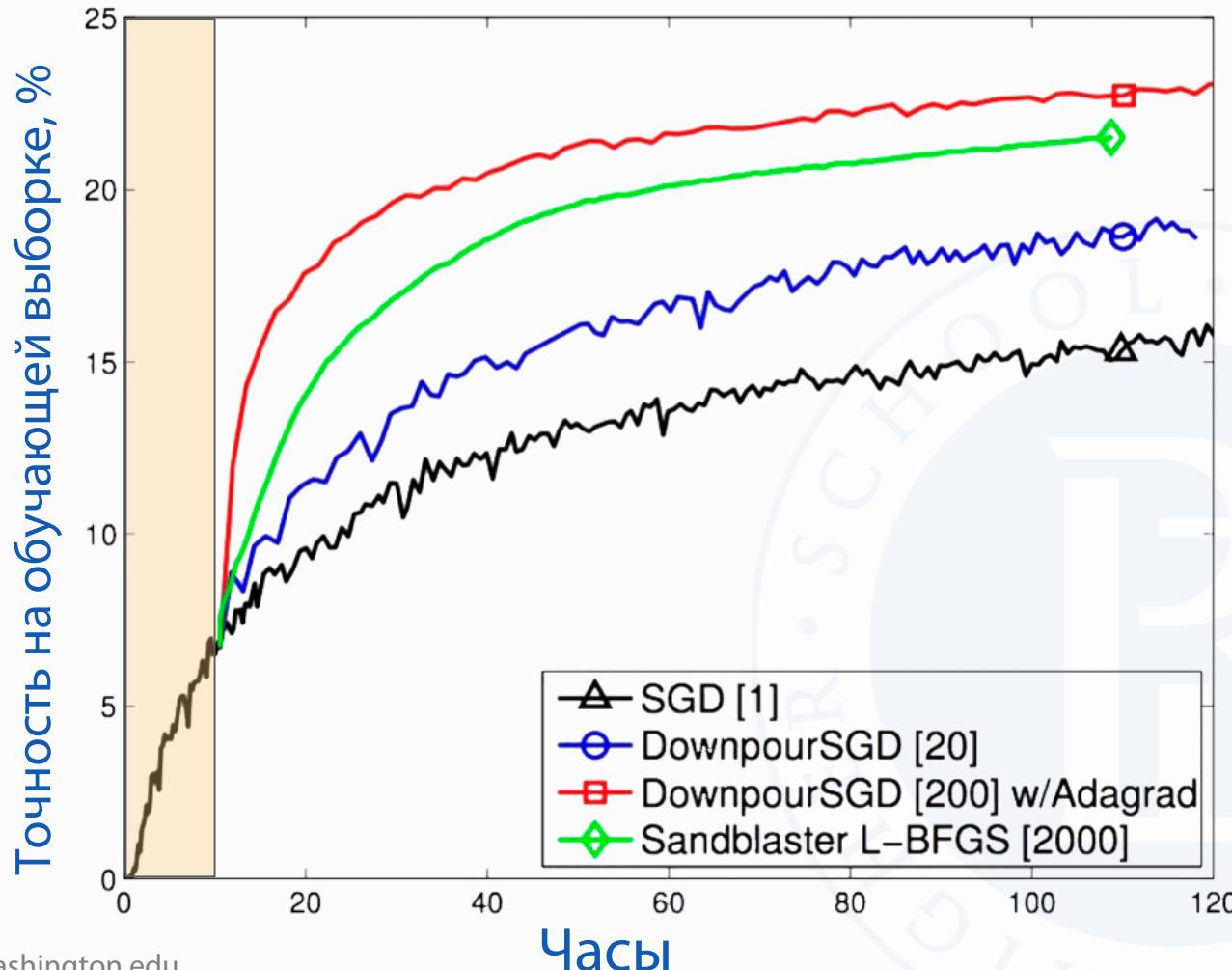
→ Обучение может не сходиться из-за расхождений асинхронного SGD



# Асинхронный SGD от Google (DistBelief)



Первая эпоха обучается синхронно



# Проблемы

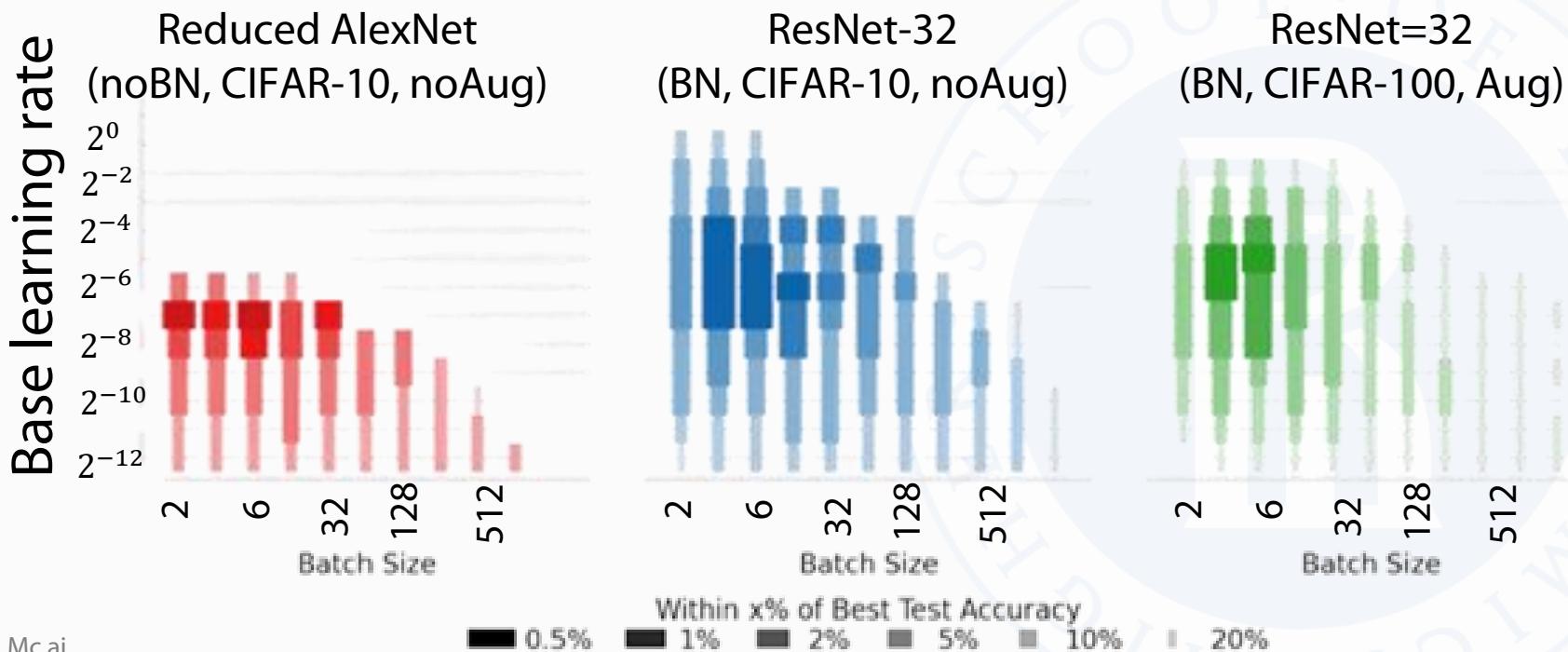
→ Размер батча



# Проблемы

## → Размер батча

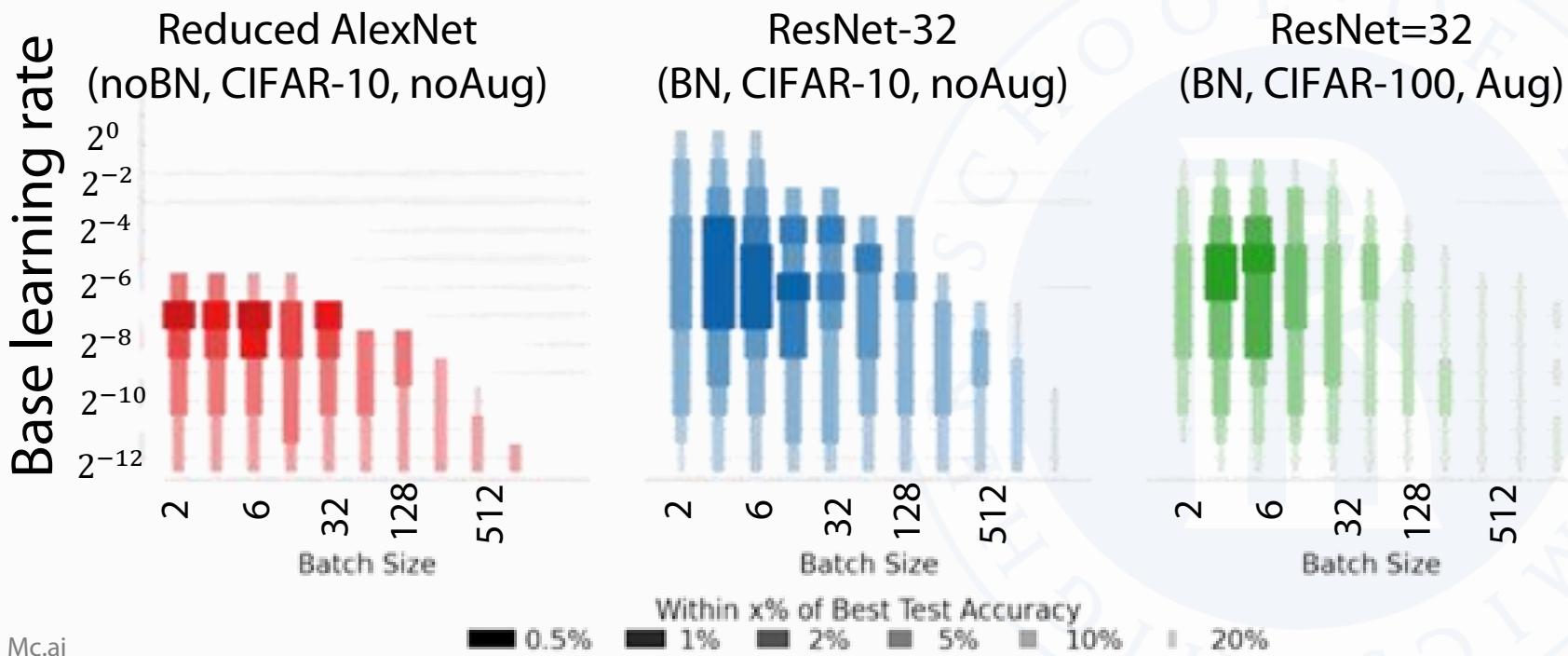
- Для утилизации GPU необходим большой батч (блок данных)



# Проблемы

## → Размер батча

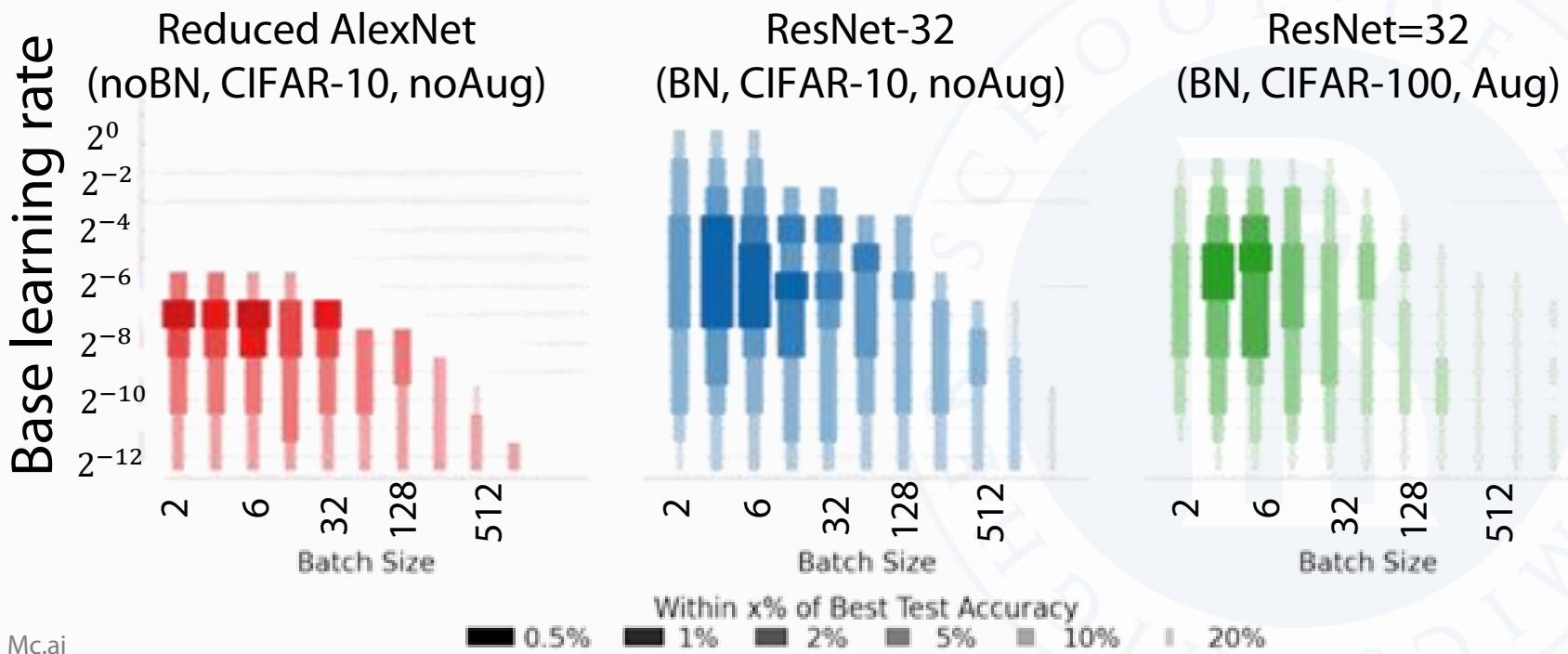
- Для утилизации GPU необходим большой батч (блок данных)
- Чем больше устройств и памяти, тем больше блок необходим



# Проблемы

## → Размер батча

- Для утилизации GPU необходим большой батч (блок данных)
- Чем больше устройств и памяти, тем больше блок необходим
- При большом блоке обучение может не сойтись



# Проблемы



## Размер батча

- Для утилизации GPU необходим большой батч (блок данных)
- Чем больше устройств и памяти, тем больше блок необходим
- При большом блоке обучение может не сойтись



## Размер модели

# Проблемы

## → Размер батча

- Для утилизации GPU необходим большой батч (блок данных)
- Чем больше устройств и памяти, тем больше блок необходим
- При большом блоке обучение может не сойтись

## → Размер модели

- Чем больше модель, тем больше градиенты

# Проблемы



## Размер батча

- Для утилизации GPU необходим большой батч (блок данных)
- Чем больше устройств и памяти, тем больше блок необходим
- При большом блоке обучение может не сойтись



## Размер модели

- Чем больше модель, тем больше градиенты
- Передача больших объектов влияет на скорость обучения

# Итоги



В этом видео мы:



изучили Parameter Server



# Итоги



В этом видео мы:

- изучили Parameter Server
- обсудили асинхронный SGD



# Итоги



В этом видео мы:

- изучили Parameter Server
- обсудили асинхронный SGD
- обсудили Distributed Delayed Proximal Gradient

# Итоги



В этом видео мы:

- изучили Parameter Server
- обсудили асинхронный SGD
- обсудили Distributed Delayed Proximal Gradient
- поговорили о проблемах распределенного обучения нейросетей

# Далее



В следующем видео поговорим о продвинутом методе подсчета синхронного SGD — [Ring-Allreduce](#), его реализации, известной как Horovod, и обсудим тонкости реализации алгоритмов распределенного обучения нейронных сетей



# **Синхронный SGD**



# План



Ring-Allreduce



# План



Ring-Allreduce



Horovod

# План



Ring-Allreduce



Horovod



Тонкости реализации алгоритмов обучения

# План



Ring-Allreduce



Horovod



Тонкости реализации алгоритмов обучения



Связанные проблемы



# Ring-Allreduce

→ Мотивация:



# Ring-Allreduce



Мотивация:

- задача Parameter Server — allreduce для обновлений

# Ring-Allreduce



Мотивация:

- задача Parameter Server — allreduce для обновлений
- обновления аддитивны

# Ring-Allreduce



Мотивация:

- задача Parameter Server — allreduce для обновлений
- обновления аддитивны
- упираемся в сеть

# Ring-Allreduce



Мотивация:

- задача Parameter Server — allreduce для обновлений
- обновления аддитивны
- упираемся в сеть



Можем ли обновлять данные на обработчиках  
децентрализованно и утилизировать сеть **оптимальнее?**

# Ring-Allreduce

- Мотивация:
  - задача Parameter Server — allreduce для обновлений
  - обновления аддитивны
  - упираемся в сеть
- Может ли обновлять данные на обработчиках децентрализованно и утилизировать сеть оптимальнее?
- Да, мы уже знаем Tree-Allreduce, Ring — альтернатива

# Ring-Allreduce

→ Суммируем данные при передаче на первых  $N - 1$  шагах

Обработчик 1

5 13

8 19

42 1

Обработчик 2

9 27

3 15

8 4

Обработчик 3

8 11

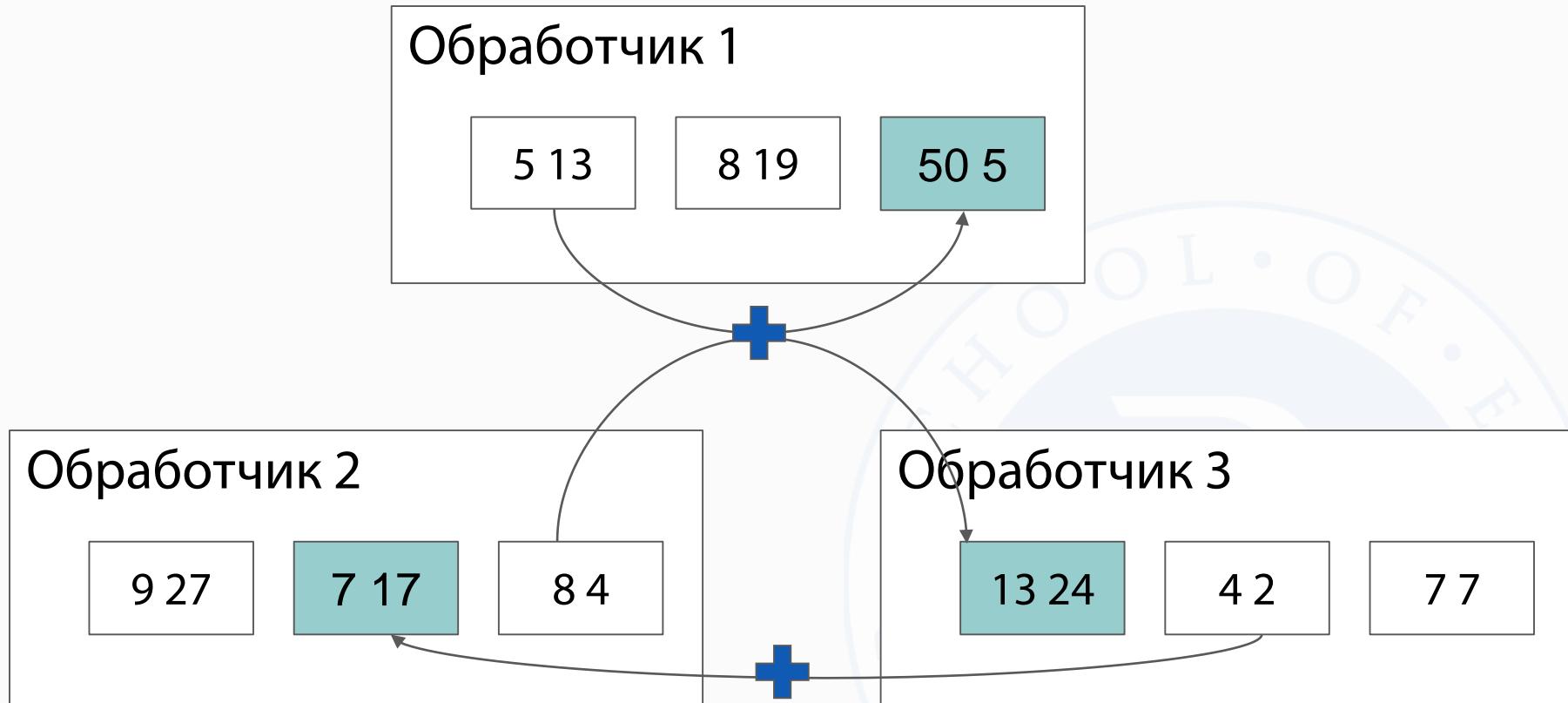
4 2

7 7

# Ring-Allreduce



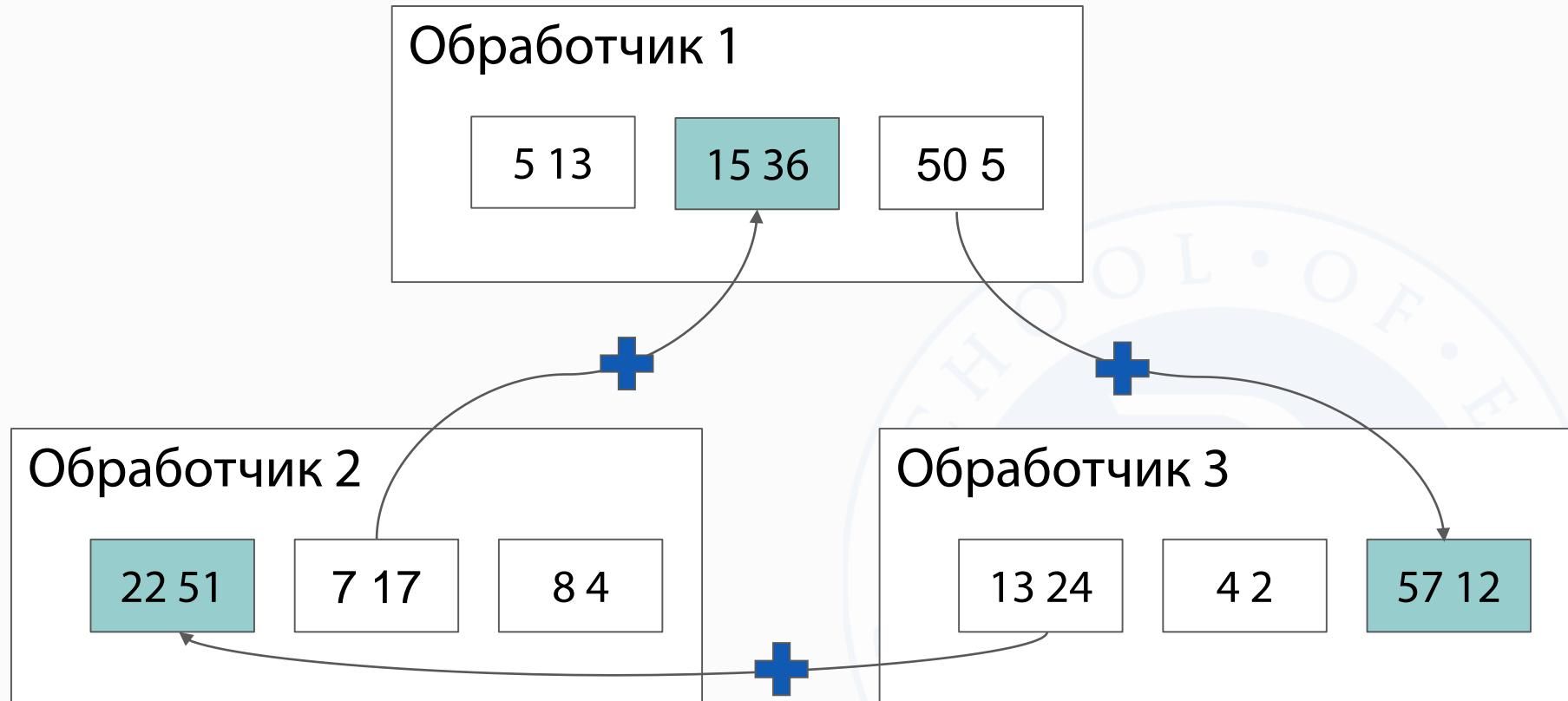
Суммируем данные при передаче на первых  $N - 1$  шагах



# Ring-Allreduce

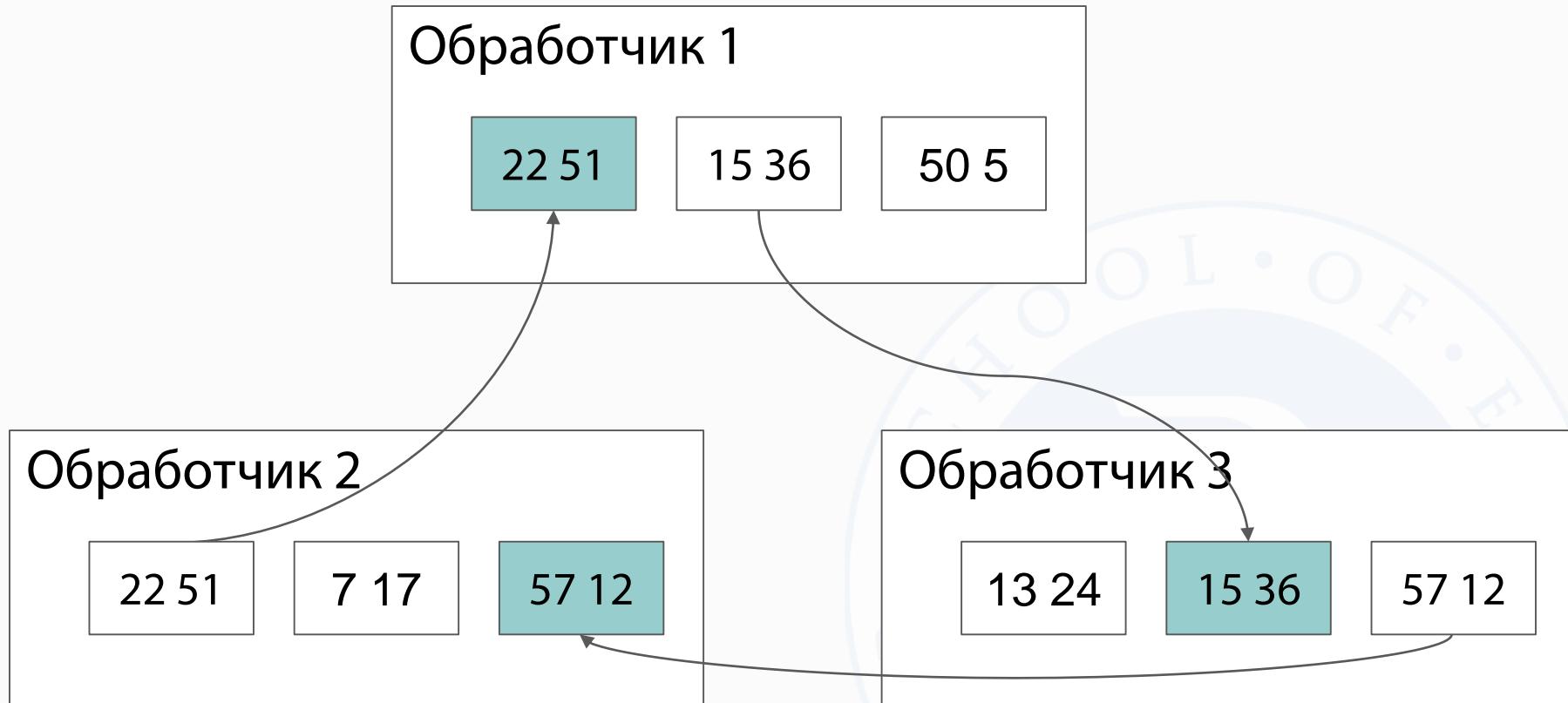


Суммируем данные при передаче на первых  $N - 1$  шагах



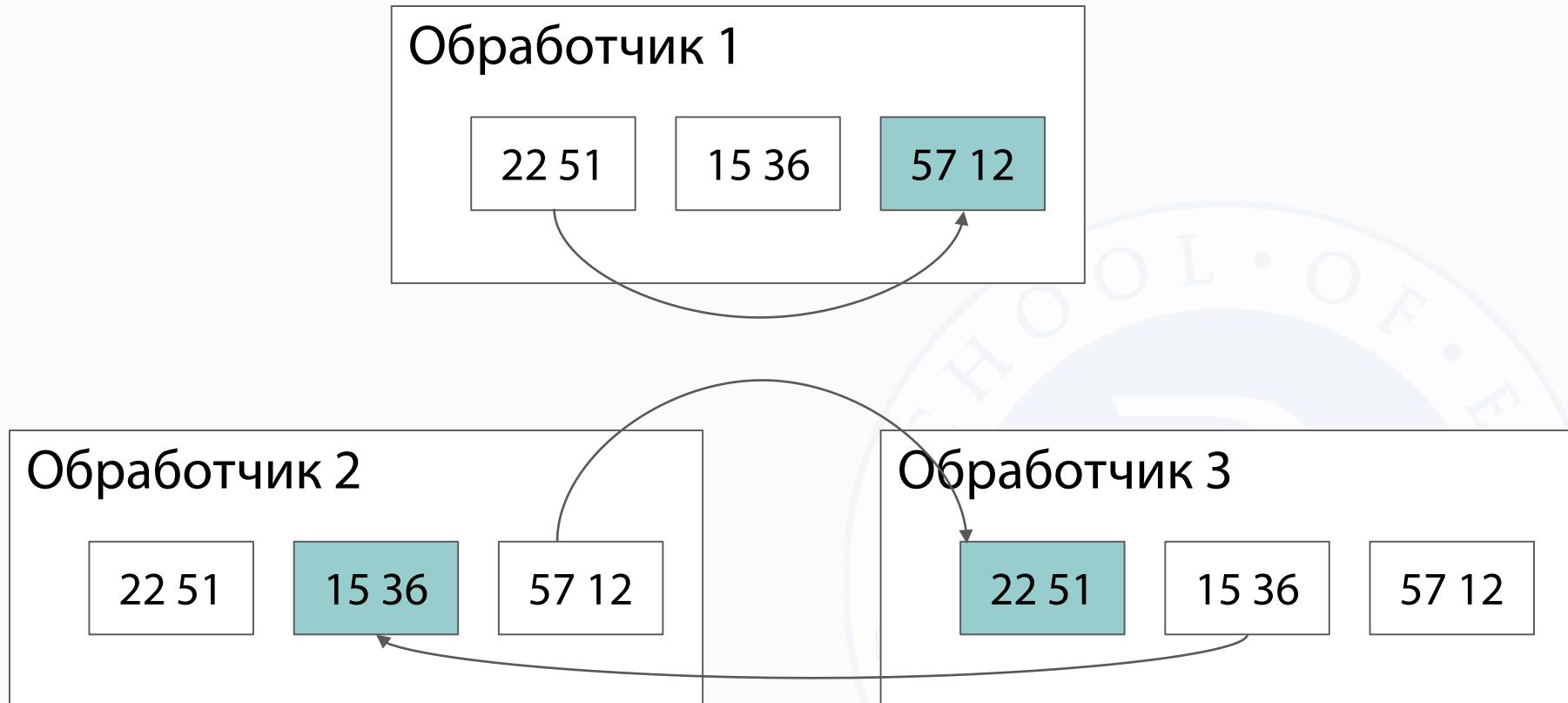
# Ring-Allreduce

→ На следующих  $N - 1$  шагах данные заменяются



# Ring-Allreduce

→ На следующих  $N - 1$  шагах данные заменяются



# Ring-Allreduce



Во всех ячейках всех узлов результат суммы исходных данных

Обработчик 1

22 51

15 36

57 12

Обработчик 2

22 51

15 36

57 12

Обработчик 3

22 51

15 36

57 12

# Ring-Allreduce

→ Является Allreduce операцией

# Ring-Allreduce

- Является Allreduce операцией
- Не требуется сервер-координатор

# Ring-Allreduce

- Является Allreduce операцией
- Не требуется сервер-координатор
- Эффективно утилизирует сеть —  $2N - 1$  передач

# Ring-Allreduce

- Является Allreduce операцией
- Не требуется сервер-координатор
- Эффективно утилизирует сеть —  $2N - 1$  передач,  
т.е. при удачном выборе размера батча количество  
сетевых взаимодействий **оптимально**

# Ring-Allreduce

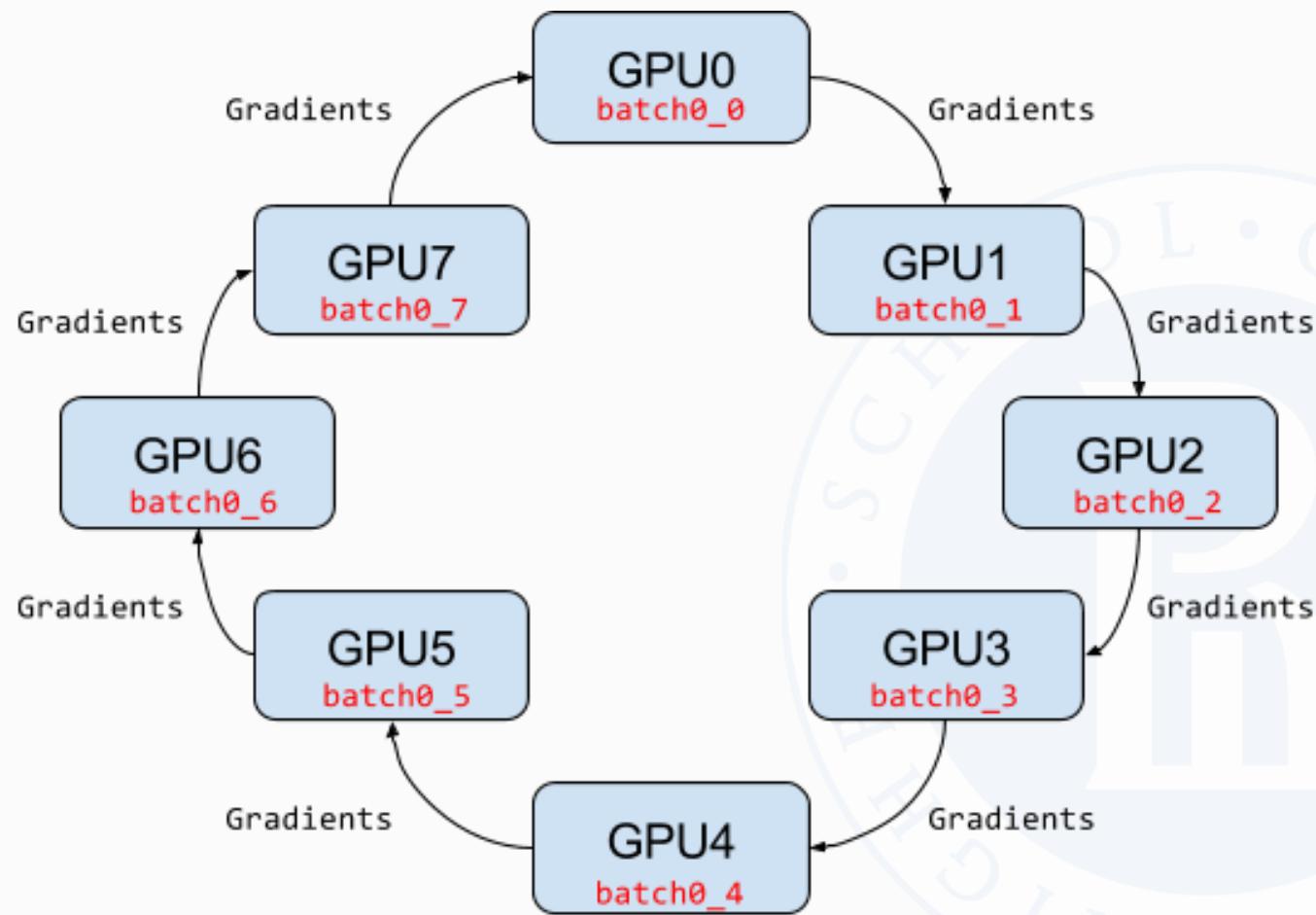
- Является Allreduce операцией
- Не требуется сервер-координатор
- Эффективно утилизирует сеть —  $2N - 1$  передач, т.е. при удачном выборе размера батча количество сетевых взаимодействий **оптимально**
- Первые  $N - 1$  шагов получатель суммирует объекты

# Ring-Allreduce

- Является Allreduce операцией
- Не требуется сервер-координатор
- Эффективно утилизирует сеть —  $2N - 1$  передач, т.е. при удачном выборе размера батча количество сетевых взаимодействий **оптимально**
- Первые  $N - 1$  шагов получатель суммирует объекты
- Последние  $N - 1$  шагов получатель заменяет данные

# Horovod

→ Алгоритм параллельного обучения нейронных сетей, основанный на подходе Ring-Allreduce



# Horovod

→ Data-Parallel



# Horovod

→ Data-Parallel

→ Ring-Allreduce



# Horovod

- Data-Parallel
- Ring-Allreduce
- Передаются градиенты

# Horovod

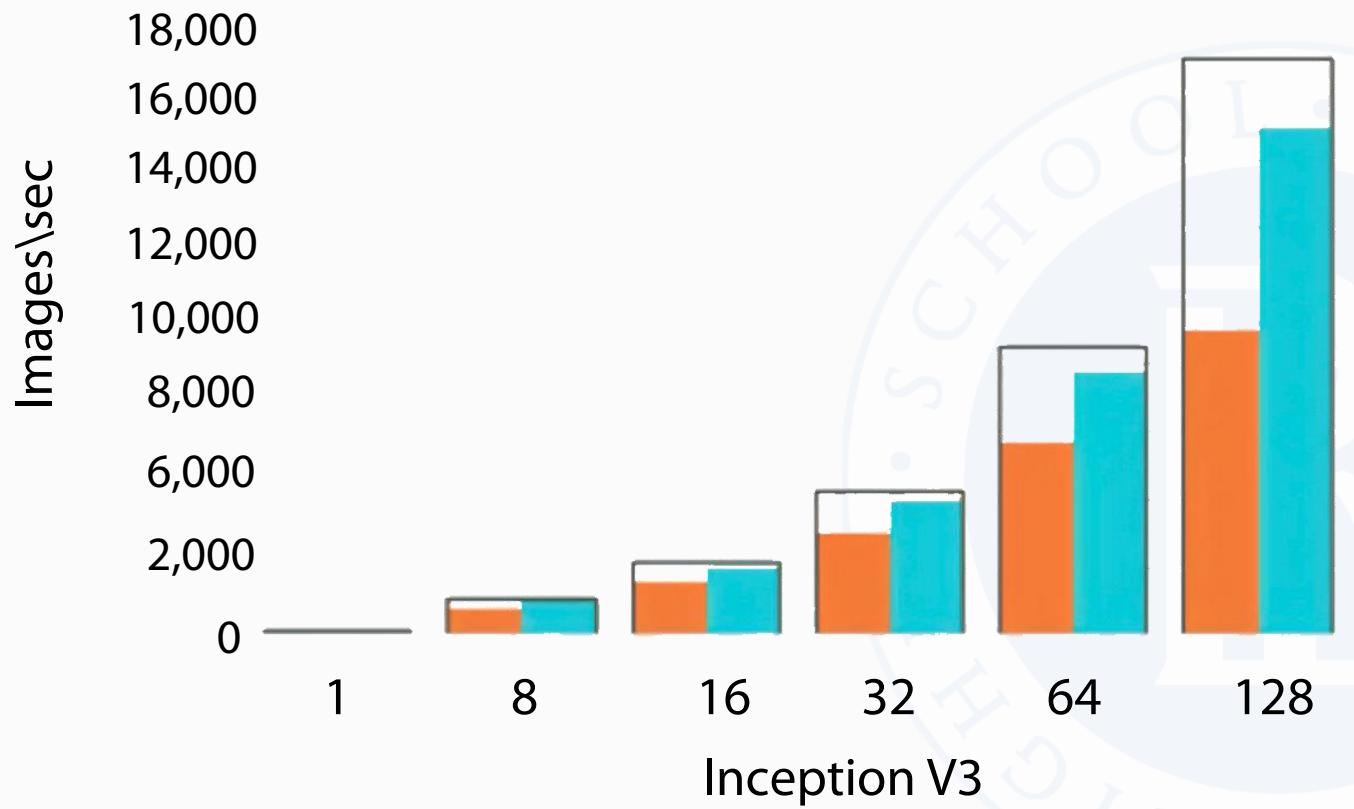
- Data-Parallel
- Ring-Allreduce
- Передаются градиенты
- Синхронный SGD



# Horovod

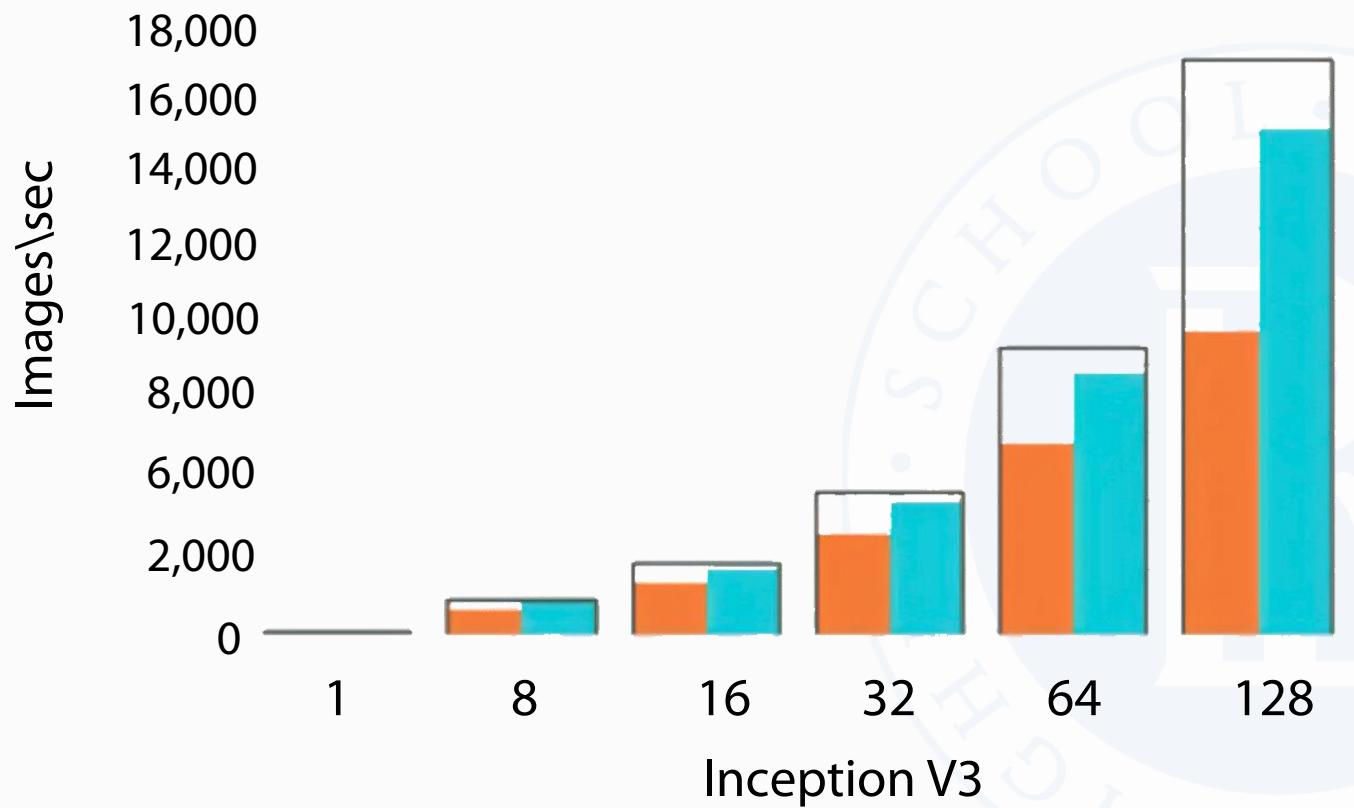


Утилизация лучше, чем Parameter Server



# Horovod

- Утилизация лучше, чем Parameter Server
- Работает с большими моделями, которые используют несколько GPU на одном узле — Model Parallel



# Тонкости реализации

Помимо MPI существуют [другие протоколы](#)



# Тонкости реализации

Помимо MPI существуют [другие протоколы](#)

- MPI — более общий интерфейс

# Тонкости реализации

Помимо MPI существуют [другие протоколы](#)

- MPI — более общий интерфейс
- Специфицированное решение — [NCCL](#)

# Тонкости реализации

Помимо MPI существуют [другие протоколы](#)

- MPI — более общий интерфейс
- Специфицированное решение — [NCCL](#)
  - Комплексная библиотека от Nvidia
  - Скрывает реализацию и выбор топологии для allreduce от пользователя

# Тонкости реализации

- PyTorch позволяет выбрать библиотеку для взаимодействия: Gloo, NCCL или MPI

# Тонкости реализации

- PyTorch позволяет выбрать библиотеку для взаимодействия: Gloo, NCCL или MPI
- Рекомендуют использовать разные библиотеки для обучения:
  - CPU — Gloo

# Тонкости реализации

- PyTorch позволяет выбрать библиотеку для взаимодействия: Gloo, NCCL или MPI
- Рекомендуют использовать разные библиотеки для обучения:
  - CPU — Gloo
  - GPU — NCCL

# Проблемы

- Существует ряд физических ограничений:
  - память на GPU

# Проблемы

→ Существует ряд физических ограничений:

- память на GPU
- мощность CPU и объем оперативной памяти

# Проблемы

→ Существует ряд физических ограничений:

- память на GPU
- мощность CPU и объем оперативной памяти
- ширина канала передачи данных

# Проблемы

→ Существует ряд физических ограничений:

- память на GPU
- мощность CPU и объем оперативной памяти
- ширина канала передачи данных

→ При увеличении мощности отдельного узла (GPU/CPU) можно обучать модели быстрее, что логично, но...

# Проблемы

→ Существует ряд физических ограничений:

- память на GPU
- мощность CPU и объем оперативной памяти
- ширина канала передачи данных

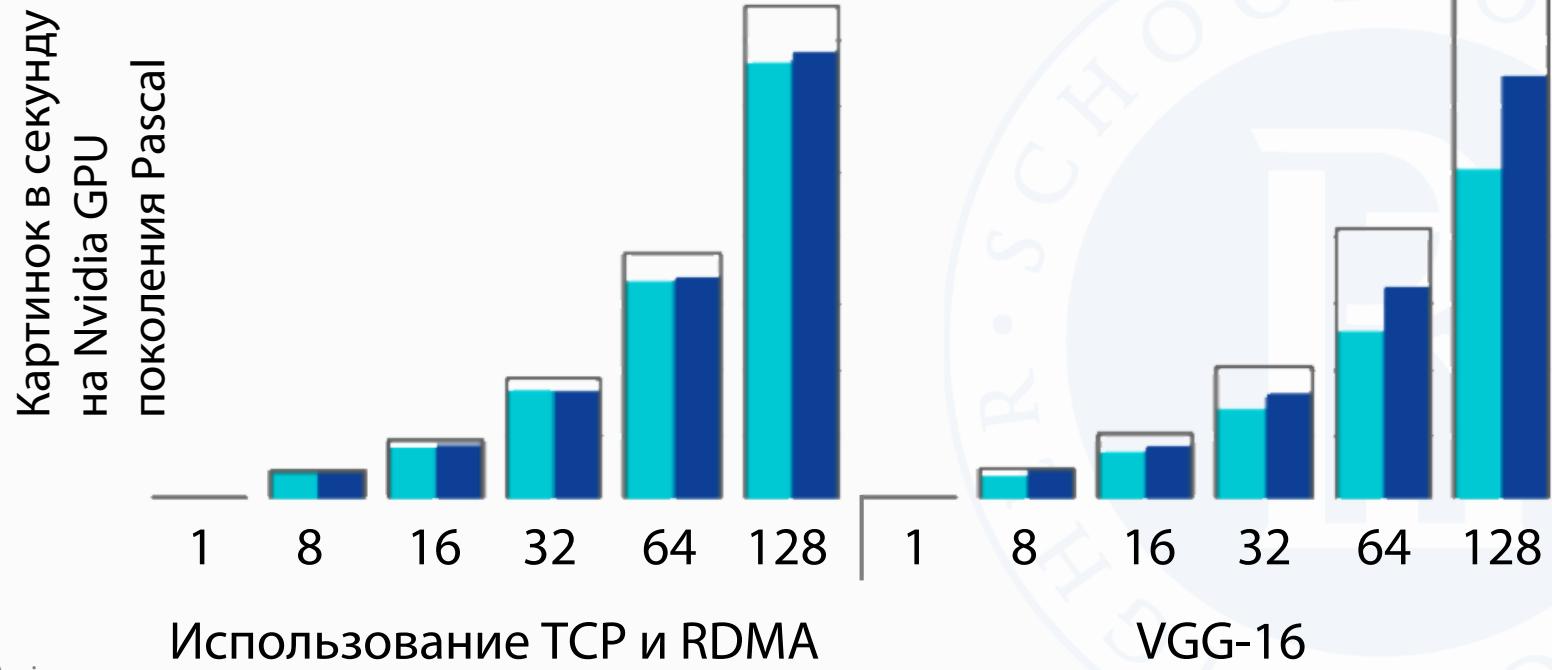
→ При увеличении мощности отдельного узла (GPU/CPU) можно обучать модели быстрее, что логично, но...

→ Параллелизм опирается на взаимодействие между обработчиками, поэтому требует хороший канал связи для передачи большого количества данных

# Ширина канала. InfiniBand



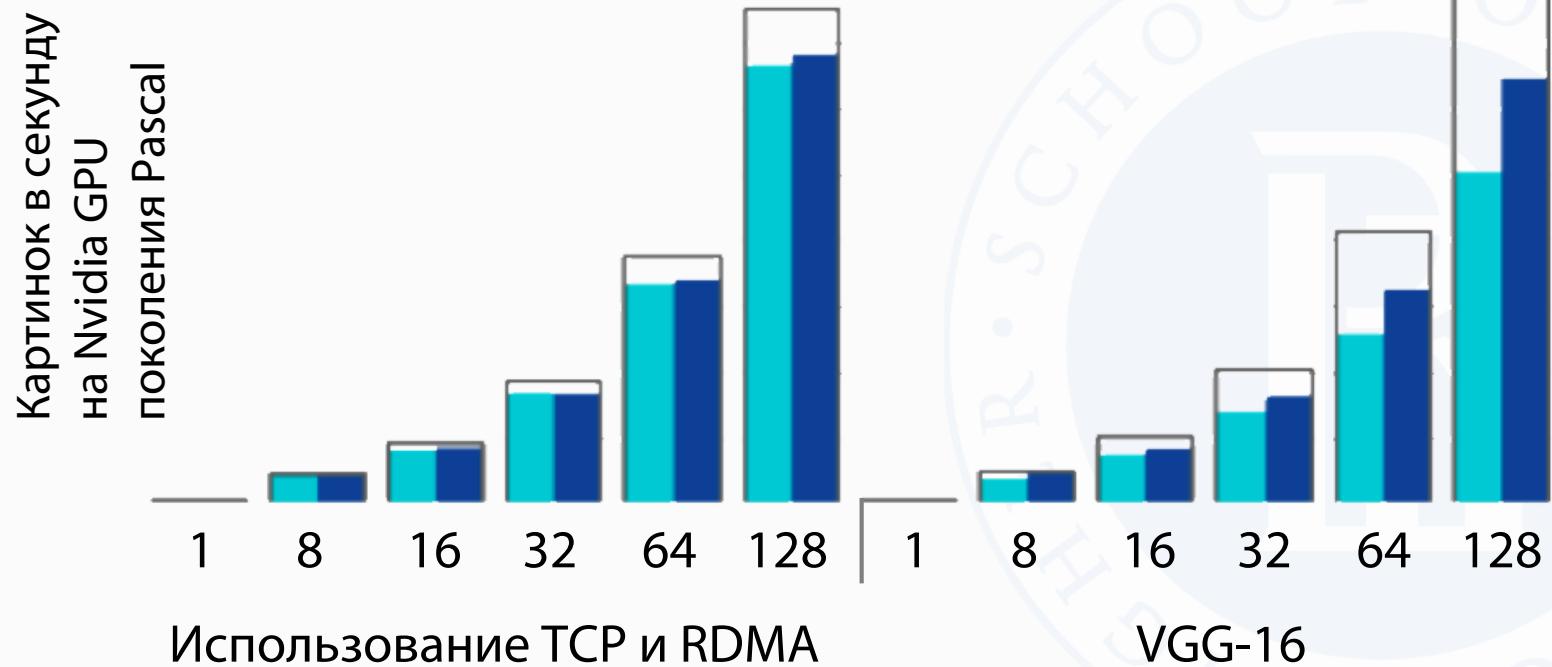
InfiniBand — сеть с большой шириной канала и низкими задержками, конкурент Ethernet



# Ширина канала. InfiniBand

→ InfiniBand — сеть с большой шириной канала и низкими задержками, конкурент Ethernet

→ Позволяет использовать RDMA  
*Remote Direct Memory Access* — доступ к памяти без использования CPU удаленного устройства



# Ширина канала. NVLink

→ NVLink — решение от NVidia для объединения GPU в кластер



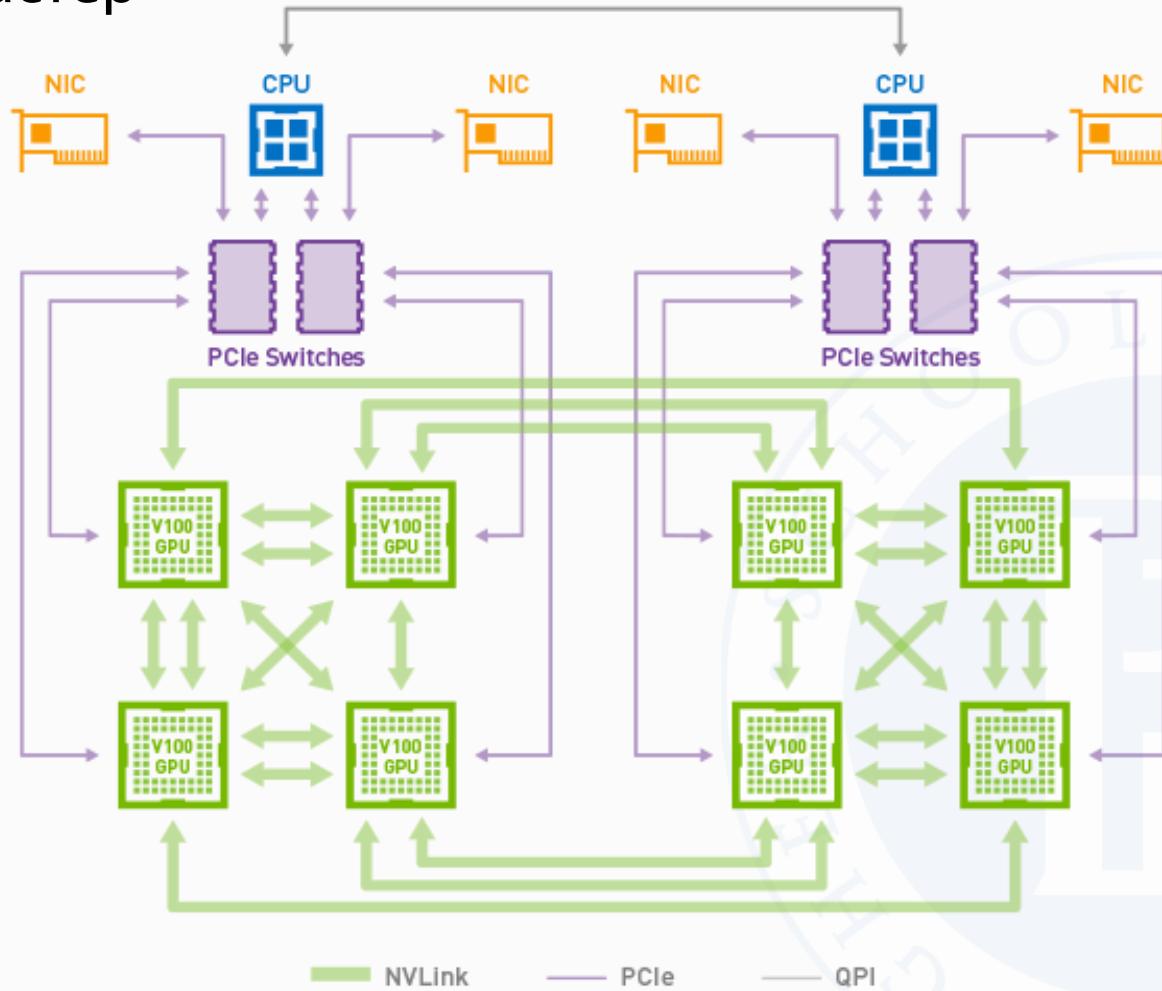
# Ширина канала. NVLink

- NVLink — решение от NVidia для объединения GPU в кластер
- Скорость — до 20 гбайт/сек



# Ширина канала. NVLink

→ NVLink — решение от NVidia для объединения GPU в кластер



# Итоги



В этом видео мы познакомились с:



Ring-Allreduce



# Итоги



В этом видео мы познакомились с:

→ Ring-Allreduce

→ Horovod



# Итоги



В этом видео мы познакомились с:



Ring-Allreduce



Horovod



тонкостями нижележащих уровней и их влиянием  
на скорость обучения

# Далее



Практическое занятие по применению Parameter Server и Horovod с использованием PyTorch



# **Перенос обучения**



# План



Задача переноса обучения



# План



Задача переноса обучения



Дообучение всей модели

# План



Задача переноса обучения



Дообучение всей модели



Дообучение последних слоев



# План



Задача переноса обучения



Дообучение всей модели



Дообучение последних слоев



Извлечение признаков



pre-trained модели



модели без учителя



# Transfer Learning



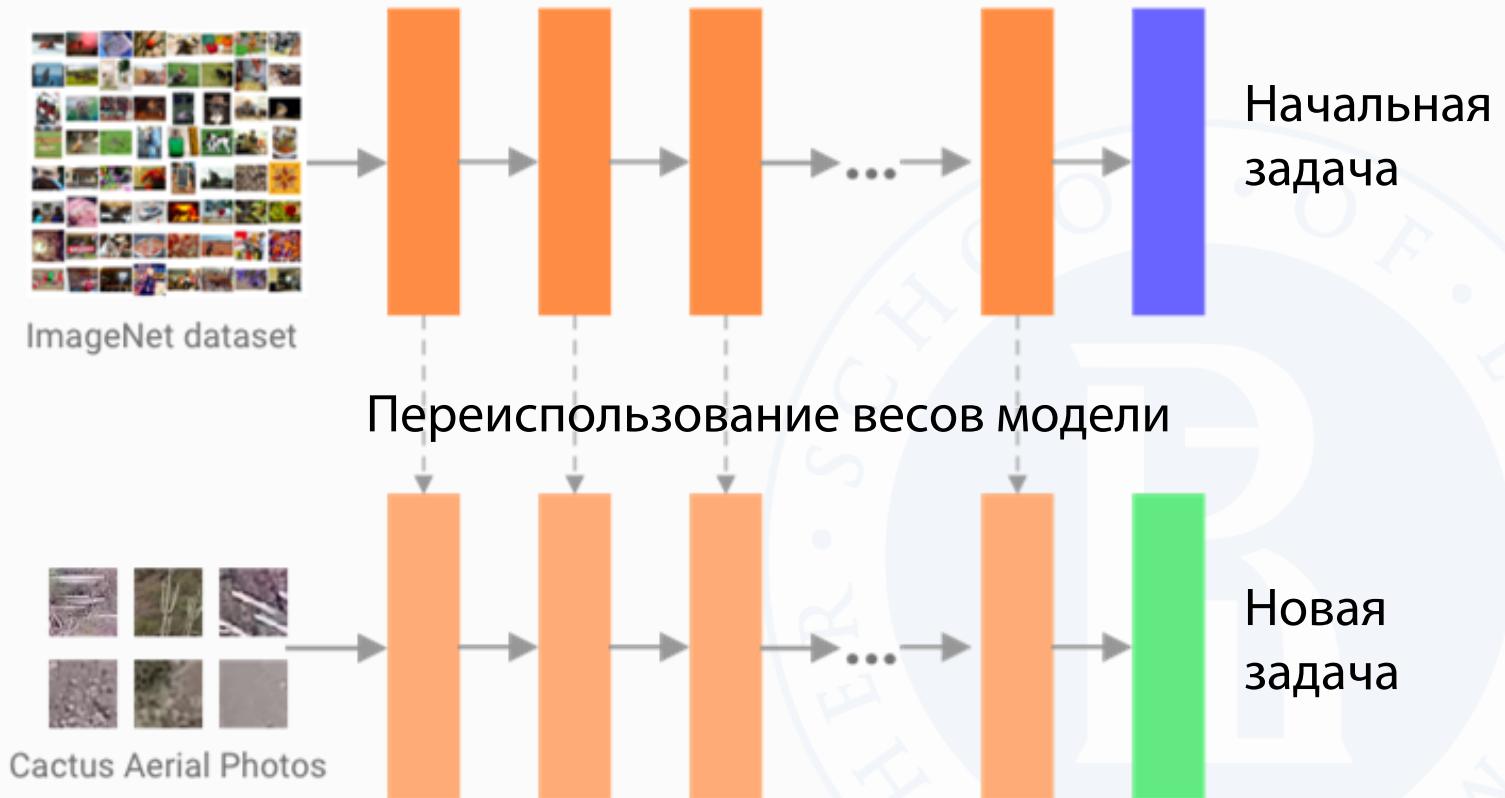
Использование знаний выученных в одной задаче  
для применения к другой задаче



# Transfer Learning



Использование знаний выученных в одной задаче  
для применения к другой задаче



# Дообучение всей модели



Инициализация модели заранее подсчитанными весами

# Дообучение всей модели

- Инициализация модели заранее подсчитанными весами
- Изменение выхода сети для решения новой задачи, инициализация последних слоев некоторыми случайными весами

# Дообучение всей модели

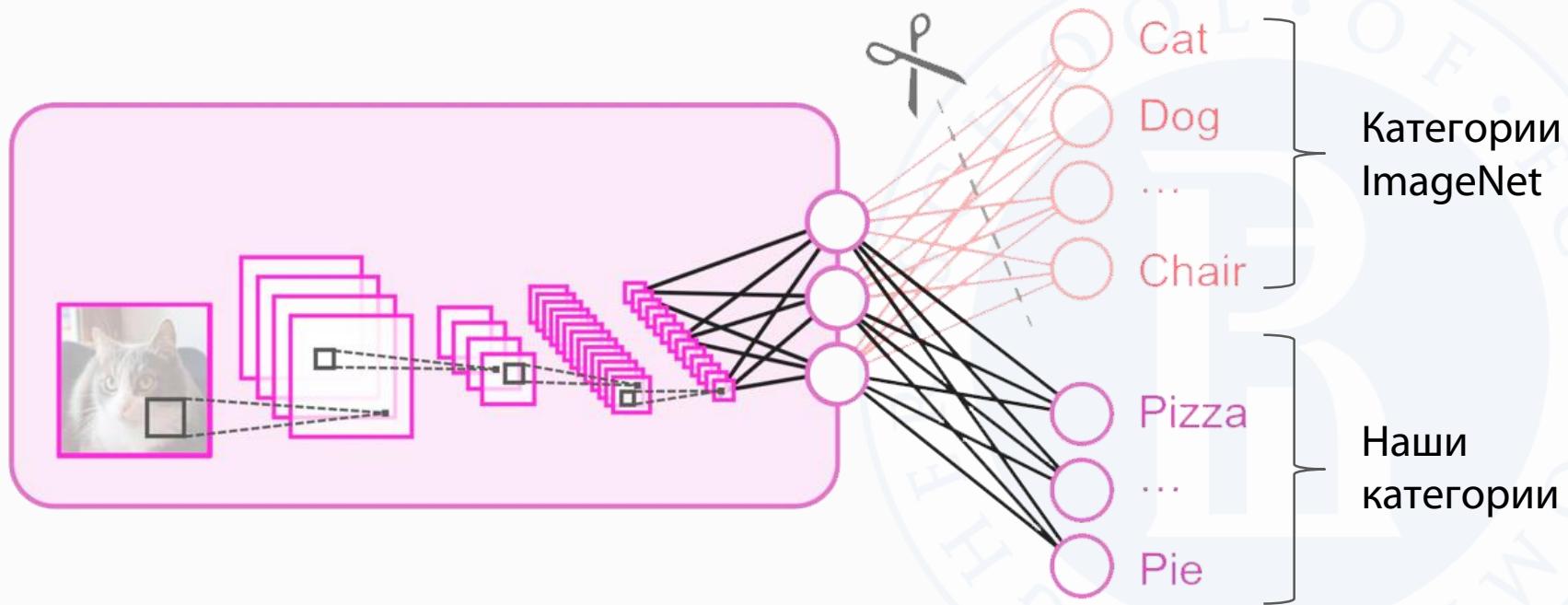
- Инициализация модели заранее подсчитанными весами
- Изменение выхода сети для решения новой задачи, инициализация последних слоев некоторыми случайными весами
- Обучение с целью сместить веса последних слоев в оптимум на новых данных

# Дообучение всей модели

- Инициализация модели заранее подсчитанными весами
- Изменение выхода сети для решения новой задачи, инициализация последних слоев некоторыми случайными весами
- Обучение с целью сместить веса последних слоев в оптимум на новых данных
- Существует несколько эвристик по использованию в зависимости от размера нового датасета

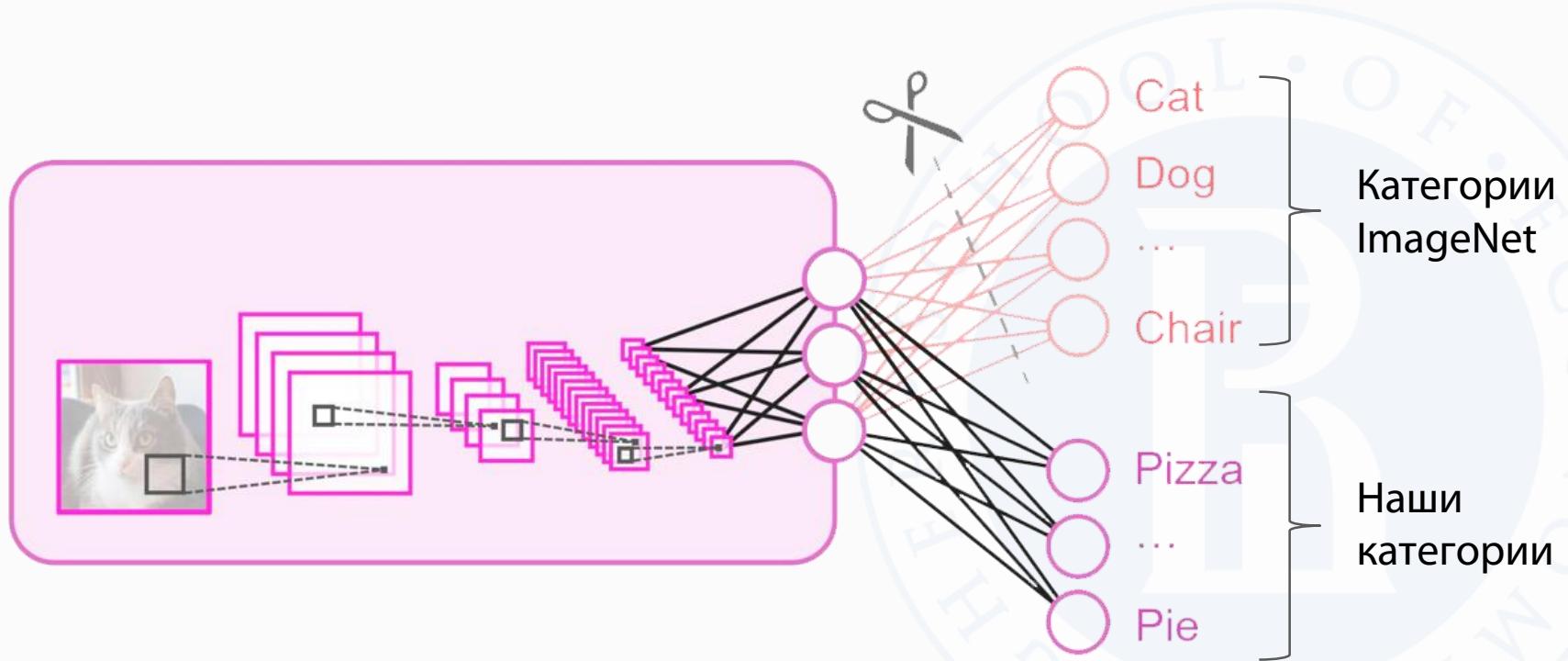
# Fine-tuning всей модели. Пример

→ Классификатор изображений по разным классам



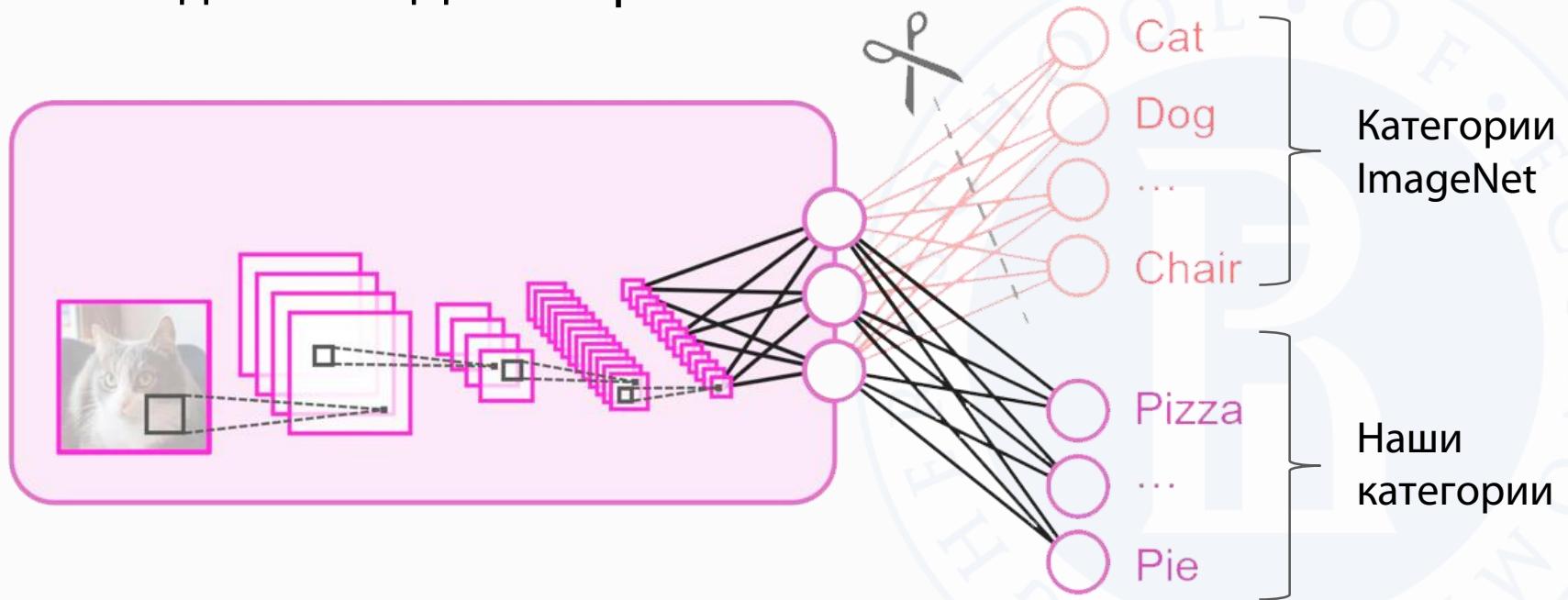
# Fine-tuning всей модели. Пример

- Классификатор изображений по разным классам
- Хотим классификатор, который бы различал виды пирогов



# Fine-tuning всей модели. Пример

- Классификатор изображений по разным классам
- Хотим классификатор, который бы различал виды пирогов
- Есть модель, обученная на ImageNet, а также датасет с видами пиццы и пирогов



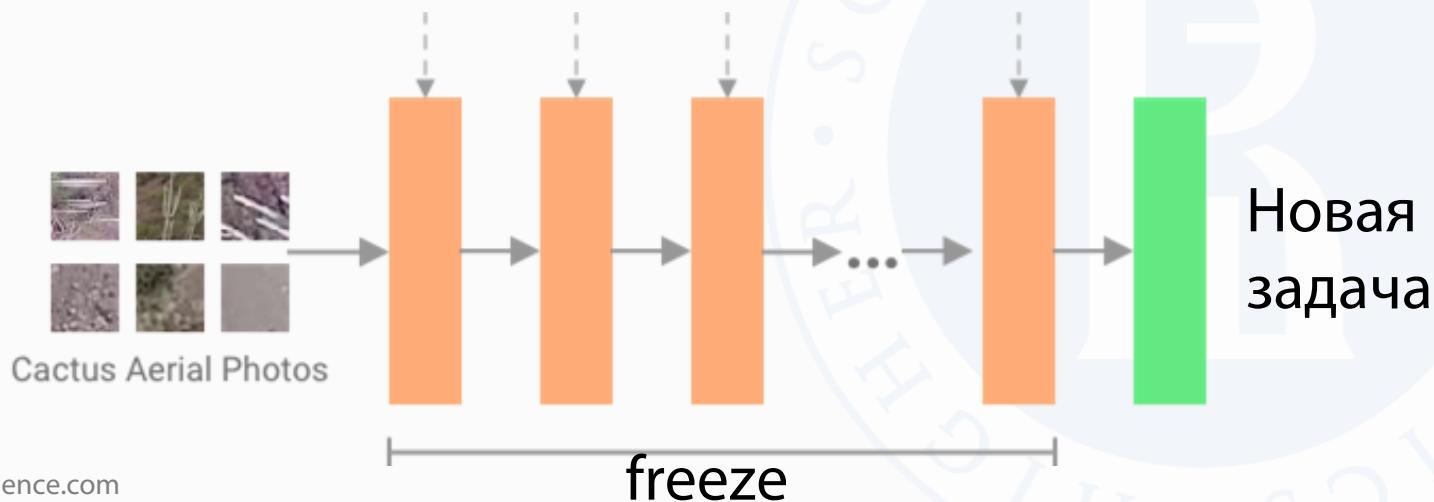
# Fine-tuning последних слоев

- Если мы не хотим менять все веса модели или у нас есть некоторые особенности, связанные с данными (об этом поговорим далее)

# Fine-tuning последних слоев

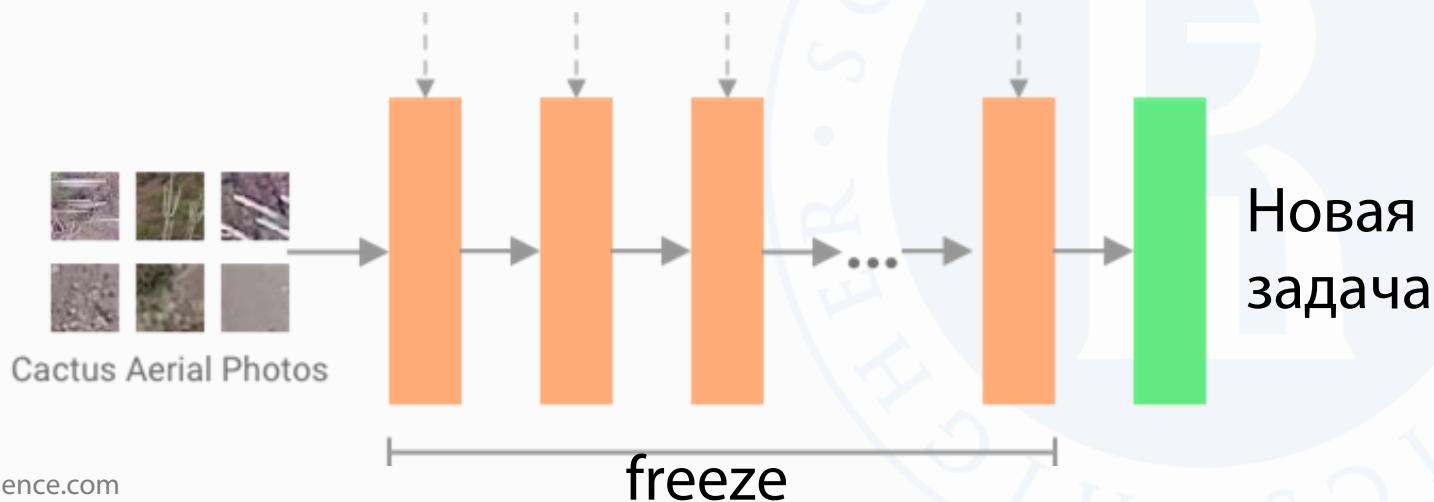


Меняем последние слои на необходимые выходы — новые классы



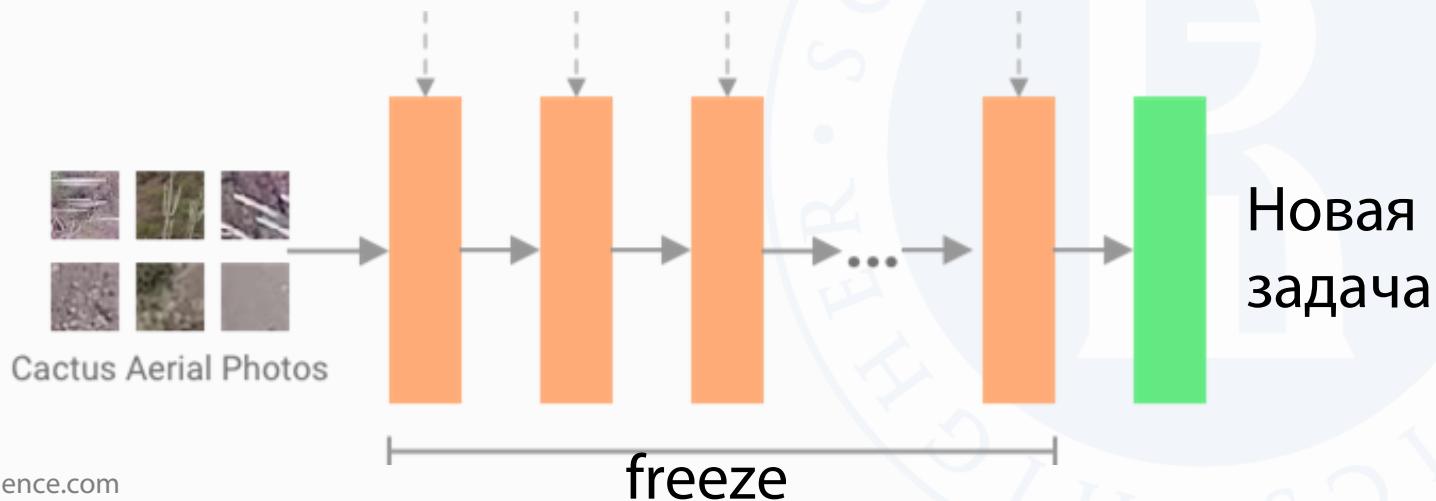
# Fine-tuning последних слоев

- Меняем последние слои на необходимые выходы — новые классы
- Замораживаем веса предыдущих слоев



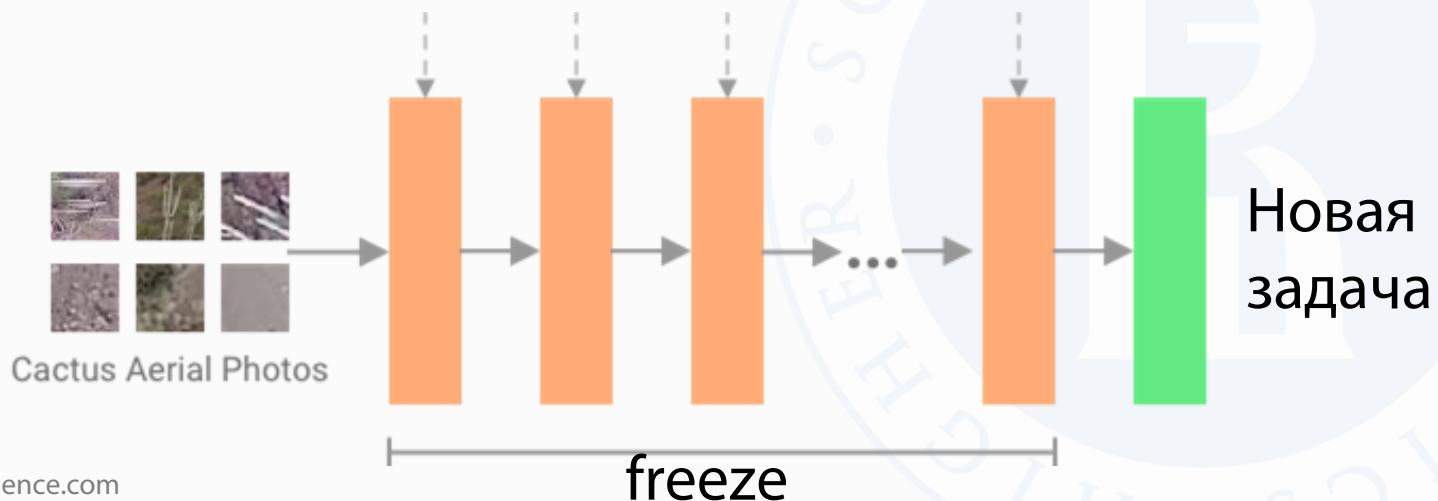
# Fine-tuning последних слоев

- Меняем последние слои на необходимые выходы — новые классы
- Замораживаем веса предыдущих слоев
- Обучаем модель на новых данных



# Fine-tuning последних слоев

- Меняем последние слои на необходимые выходы — новые классы
- Замораживаем веса предыдущих слоев
- Обучаем модель на новых данных
  - при подсчете градиентов будет вычисляться только часть для незамороженных слоев, однако проход вперед все еще потребует полного вычисления



# Fine-tuning. Эвристики

→ В зависимости от объема и характеристик новых данных:

Объем / Хар-ки	Похожи на исходные	Не похожи на исходные
Малый объем		
Большой объем		

# FT. Объем

→ Данных мало



# FT. Объем

→ **Данных мало** — в случае больших моделей есть риск переобучиться. Есть смысл обучать простую модель (например, линейную), а нейросеть использовать в качестве дополнительных факторов

# FT. Объем

- **Данных мало** — в случае больших моделей есть риск переобучиться. Есть смысл обучать простую модель (например, линейную), а нейросеть использовать в качестве дополнительных факторов
- **Данных много**

# FT. Объем

- **Данных мало** — в случае больших моделей есть риск переобучиться. Есть смысл обучать простую модель (например, линейную), а нейросеть использовать в качестве дополнительных факторов
- **Данных много** — можно обучать новую сверточную сеть или дообучать существующую

# FT. Характеристики

→ Данные похожи



# FT. Характеристики

→ Данные похожи — исходная сеть скорее всего неплохо отражает факторы нового датасета на финальных слоях

# FT. Характеристики

→ Данные похожи — исходная сеть скорее всего неплохо отражает факторы нового датасета на финальных слоях

Можно использовать базовую модель как источник факторов — не дообучать всю модель, а зафиксировать нижние слои

# FT. Характеристики

- Данные похожи — исходная сеть скорее всего неплохо отражает факторы нового датасета на финальных слоях
- Данные не похожи

# FT. Характеристики

- **Данные похожи** — исходная сеть скорее всего неплохо отражает факторы нового датасета на финальных слоях
- **Данные не похожи** — исходная сеть на последних слоях не содержит релевантной информации, но начальные слои могут хранить не специфичные для исходного датасета факторы, которые могут быть полезны

# FT. Характеристики

- **Данные похожи** — исходная сеть скорее всего неплохо отражает факторы нового датасета на финальных слоях
  
- **Данные не похожи** — исходная сеть на последних слоях не содержит релевантной информации, но начальные слои могут хранить не специфичные для исходного датасета факторы, которые могут быть полезны
  - Возможно использование дообучения всей модели

# Fine-tuning. Эвристики

→ В зависимости от объема и характеристик новых данных:

Объем / Хар-ки	Похожи на исходные	Не похожи на исходные
Малый объем	Простая модель на финальных слоях сети	Простая модель на ранних слоях сети
Большой объем	Дообучение финальных слоев, нет риска переобучения	Новая модель или дообучение всей модели

# Feature Extraction

- Промежуточное представление объекта в модели (embeddings на внутренних слоях) может быть использовано как некоторое представление объекта в пространстве меньшей размерности, чем исходный объект

# Feature Extraction

- Промежуточное представление объекта в модели (*embeddings* на внутренних слоях) может быть использовано как некоторое представление объекта в пространстве меньшей размерности, чем исходный объект
- Полученные *embeddings* можно обогащать сторонними данными и строить модели меньшего размера, чем исходная сеть

# Feature Extraction

- Промежуточное представление объекта в модели (*embeddings* на внутренних слоях) может быть использовано как некоторое представление объекта в пространстве меньшей размерности, чем исходный объект
- Полученные *embeddings* можно обогащать сторонними данными и строить модели меньшего размера, чем исходная сеть
- Дообучение с замораживанием слоев сети в некотором смысле является извлечением факторов

# Feature Extraction



Какие модели могут подойти для использования?

# Feature Extraction

- Какие модели могут подойти для использования?
- На самом деле, многие модели строят внутреннее представление
- Но насколько хорошо это представление?

# Feature Extraction

- Какие модели могут подойти для использования?
- На самом деле, многие модели строят внутреннее представление
- Но насколько хорошо это представление?
- Разделим все модели на два подхода:

Supervised

Unsupervised

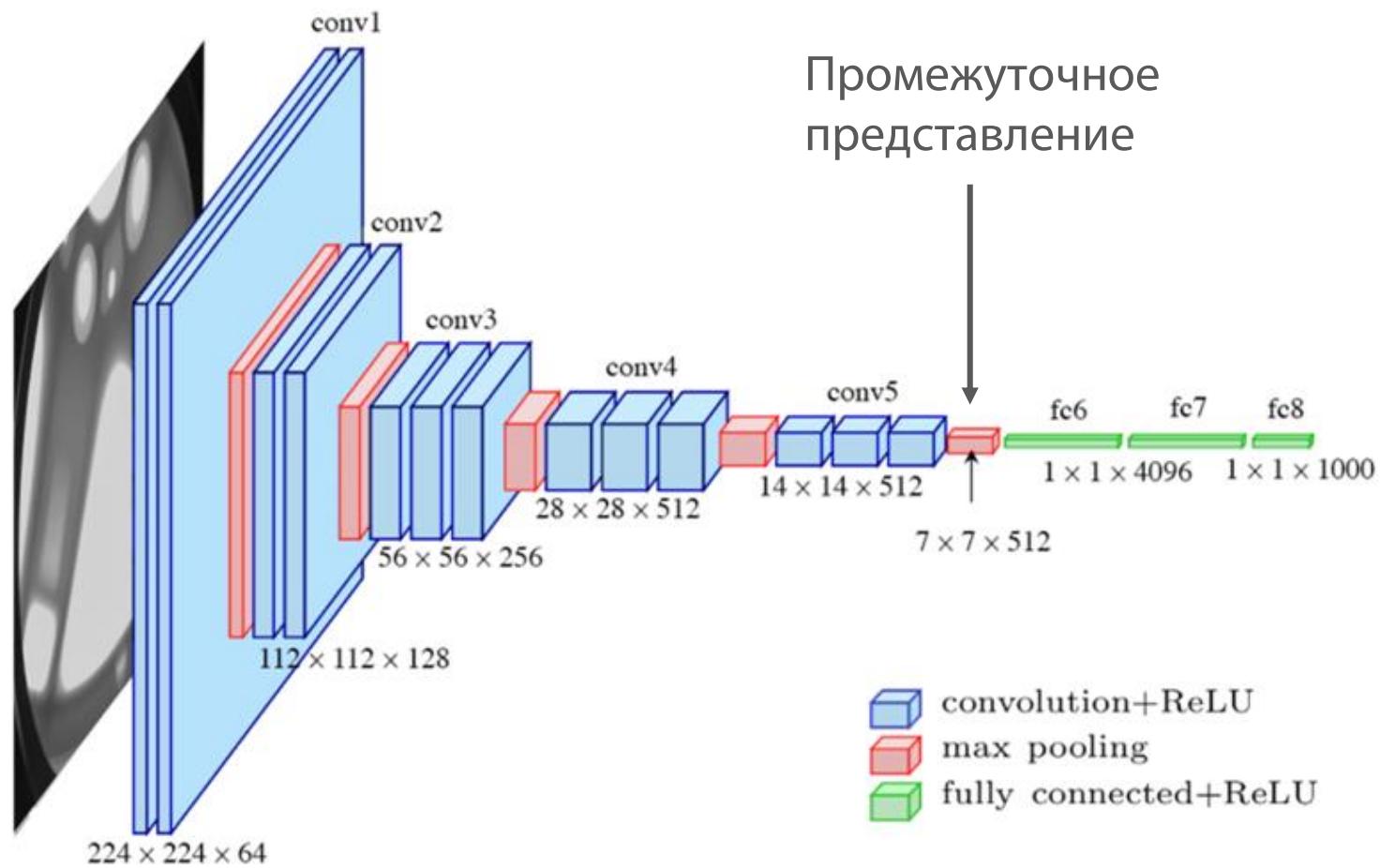
# Supervised модели

→ Это модели, которые обучаются на некоторый заданный размеченный пул объектов

→ Пример таких моделей:

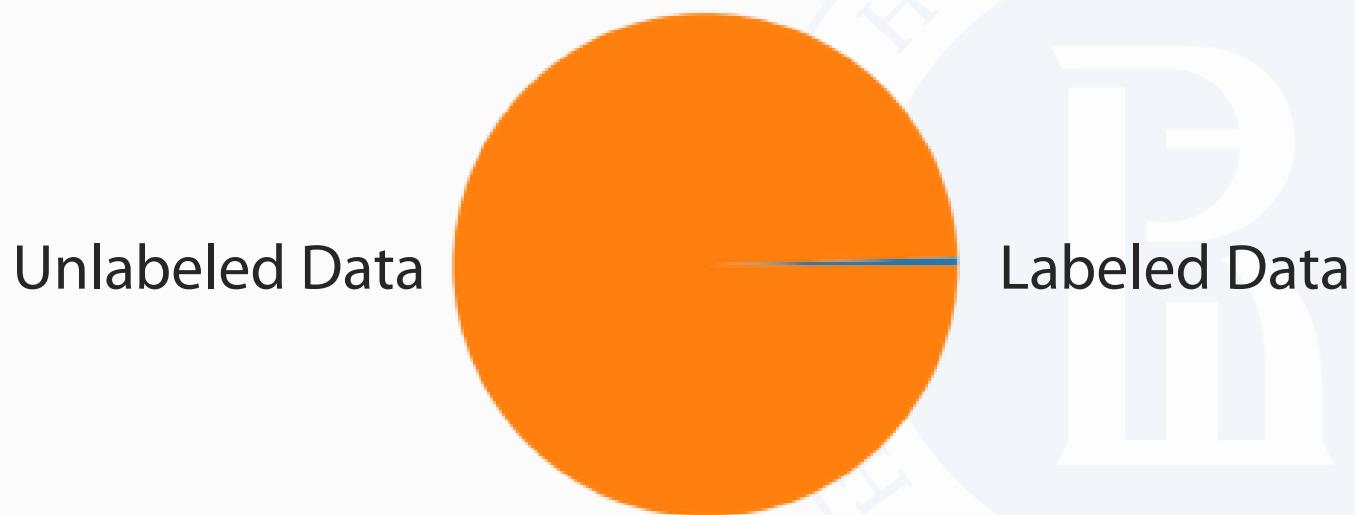
- VGG16, обученный на ImageNet — является хорошим примером модели, которая строит достаточно сильное внутреннее представление

# VGG16. Feature Extraction



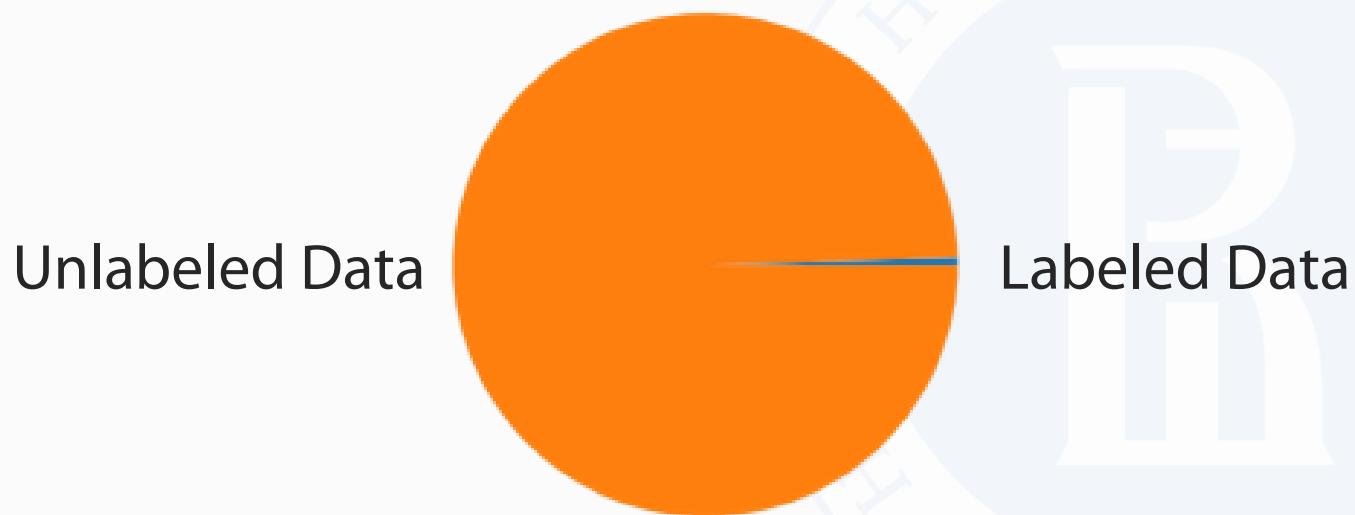
# Unsupervised модели

→ Среди таких моделей выделим два интересующих нас подхода:



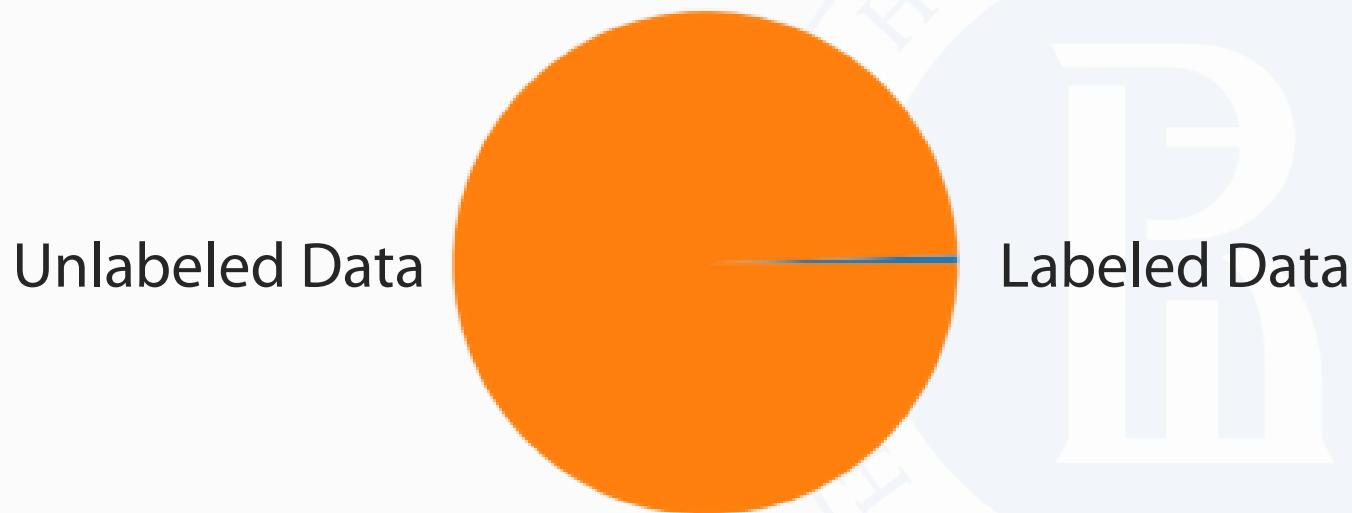
# Unsupervised модели

- Среди таких моделей выделим два интересующих нас подхода:
- Autoencoders — как представитель unsupervised



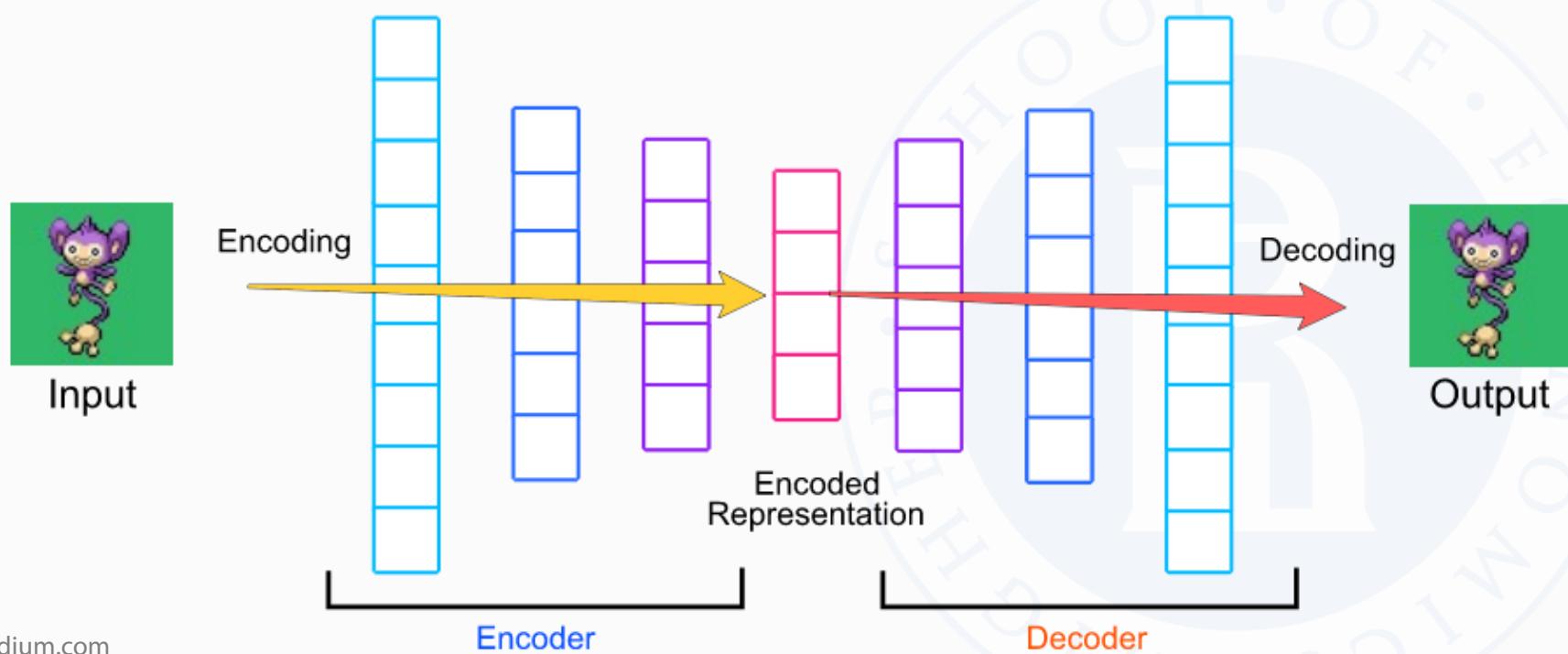
# Unsupervised модели

- Среди таких моделей выделим два интересующих нас подхода:
- Autoencoders — как представитель unsupervised
  - Contrastive learning — как разновидность self-supervised



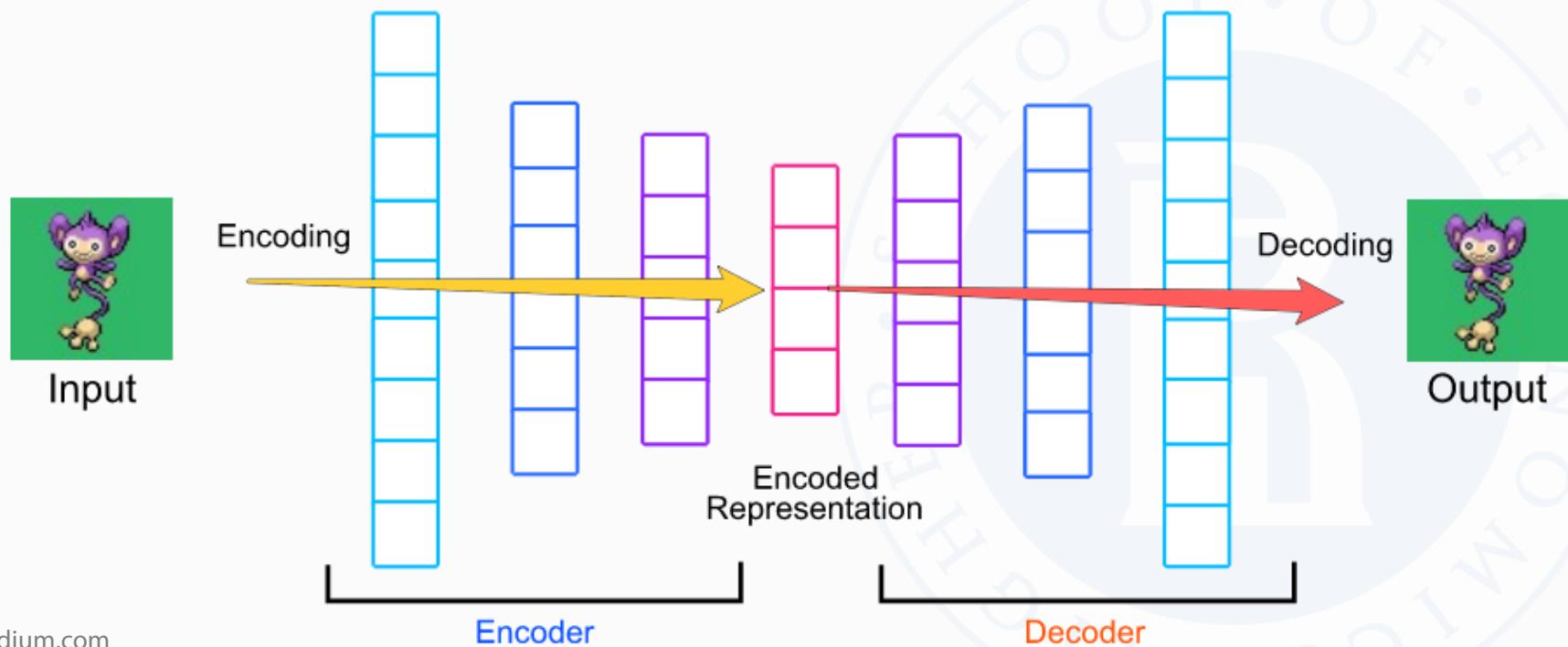
# Autoencoders

→ Модель на выходе должна получить исходный объект, имея во внутренних слоях представление меньшей размерности, чем объект на входе



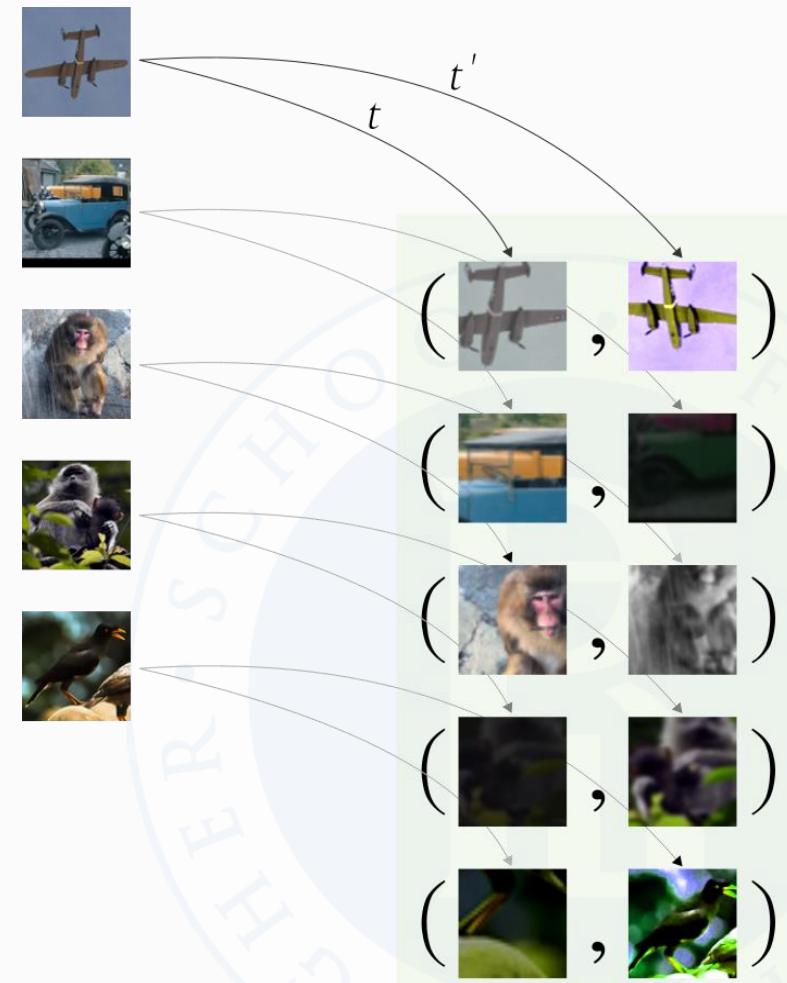
# Autoencoders

- Модель на выходе должна получить исходный объект, имея во внутренних слоях представление меньшей размерности, чем объект на входе
- Представление может быть не очень хорошим



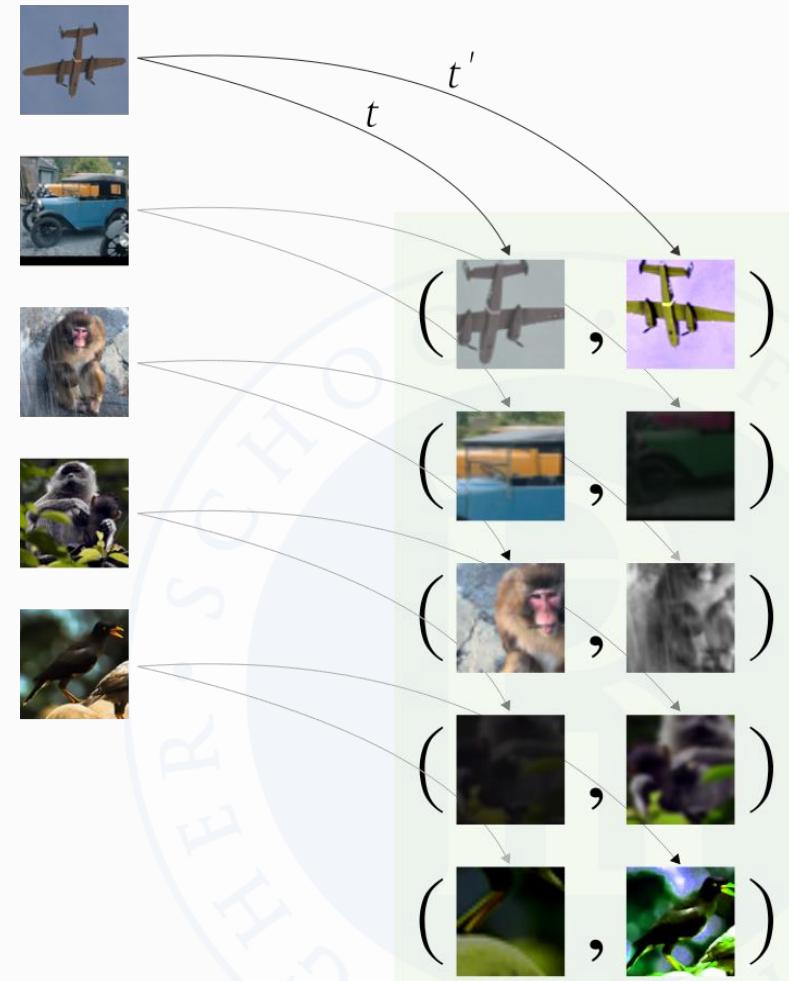
# Contrastive self-supervised learning

→ Модель учится решать задачу «похожи ли объекты»



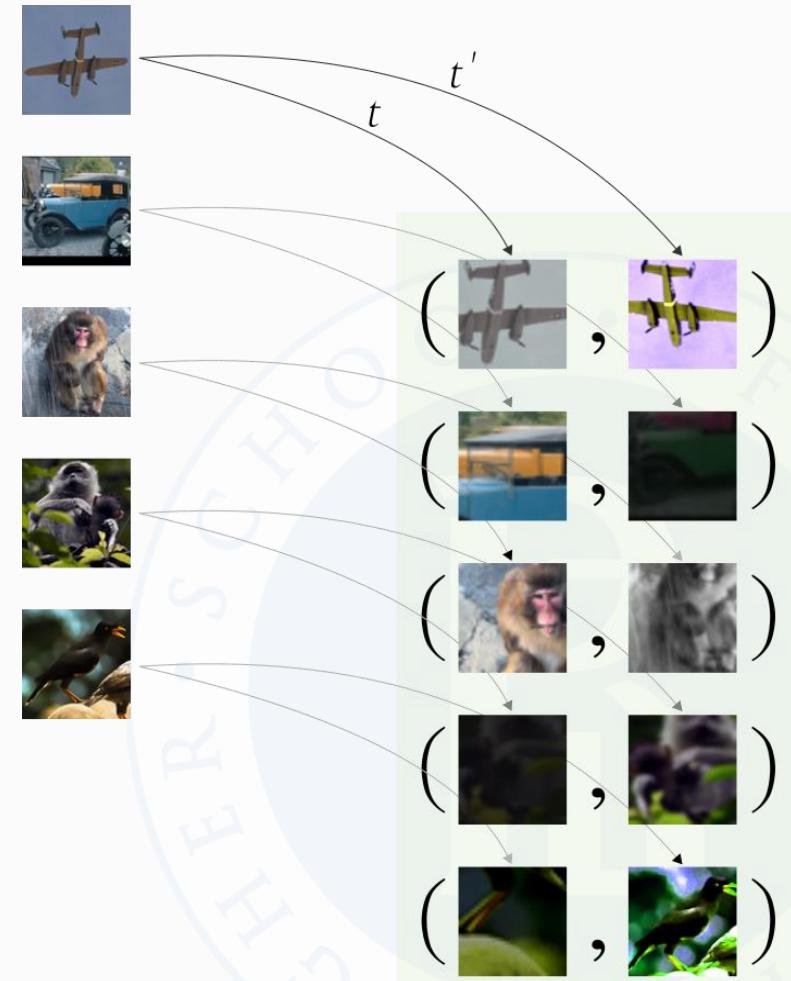
# Contrastive self-supervised learning

- Модель учится решать задачу «похожи ли объекты»
- Модель решает задачу разделения различных объектов, внутреннее представление сохраняет это свойство



# Contrastive self-supervised learning

- Модель учится решать задачу «похожи ли объекты»
- Модель решает задачу разделения различных объектов, внутреннее представление сохраняет это свойство
- Хорошо проявляют себя в задаче feature extraction



# Pre-trained bases

- Специальный подход — сочетание unsupervised стадии предобучения с supervised дообучением

# Pre-trained bases

- Специальный подход — сочетание unsupervised стадии предобучения с supervised дообучением
- GPT-3 и BERT — представители такого подхода в NLP

# Pre-trained bases

- Специальный подход — сочетание unsupervised стадии предобучения с supervised дообучением
- GPT-3 и BERT — представители такого подхода в NLP
- ViT — представитель в CV

# Pre-trained bases

- Специальный подход — сочетание unsupervised стадии предобучения с supervised дообучением
- GPT-3 и BERT — представители такого подхода в NLP
- ViT — представитель в CV
- Учатся «понимать» пространство объектов на стадии pre-training, затем дообучаются решать конкретную задачу

# Pre-trained bases

- Специальный подход — сочетание unsupervised стадии предобучения с supervised дообучением
- GPT-3 и BERT — представители такого подхода в NLP
- ViT — представитель в CV
- Учатся «понимать» пространство объектов на стадии pre-training, затем дообучаются решать конкретную задачу
- Подробнее — на практическом занятии

# Feature Extraction. Пример

→ Найти все фотографии, содержащие одного человека, имея модель, которая умеет находить лицо на фото



# Итоги



В этом видео мы:



Поговорили про задачу переноса обучения



# Итоги



В этом видео мы:

- Поговорили про задачу переноса обучения
- Изучили дообучение сетей



# Итоги



В этом видео мы:

- Поговорили про задачу переноса обучения
- Изучили дообучение сетей
- Разобрали подходы к извлечению факторов

# Далее



Практическое занятие по переносу обучения, дообучим VGG19 под свой датасет, научимся извлекать внутреннее представление, а также дообучим большую модель ViT для решения конкретной задачи

