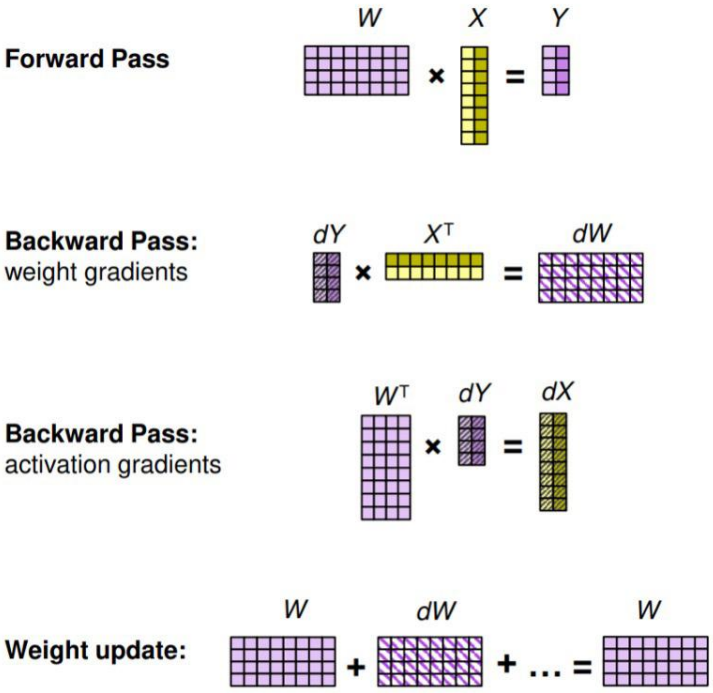
Распределенное обучение нейросетей

Артем Малько, БПМИ 213

План доклада

- Как мы можем распараллелить нейросеть?
- Способы и что они нам дают:
  - Data Parallel
  - Model Parallel
  - Tensor Parallel
- Примеры использования

Основные шаги в backprop



я жду, пока обучится нейронка…

Формальная постановка задачи

- Будем использовать датасет ImageNET
- Также возьмем SGD:

$$w_{t+1} = w_t - \eta \frac{1}{n} \sum_{x \in \mathcal{B}} \nabla l(x, w_t).$$

- Делаем Warmup – Learning Rate сначала маленький, постепенно увеличивается
- Learning Rate меняем по правилу ниже

> ***Linear Scaling Rule:*** *When the minibatch size is multiplied by k, multiply the learning rate by k.*

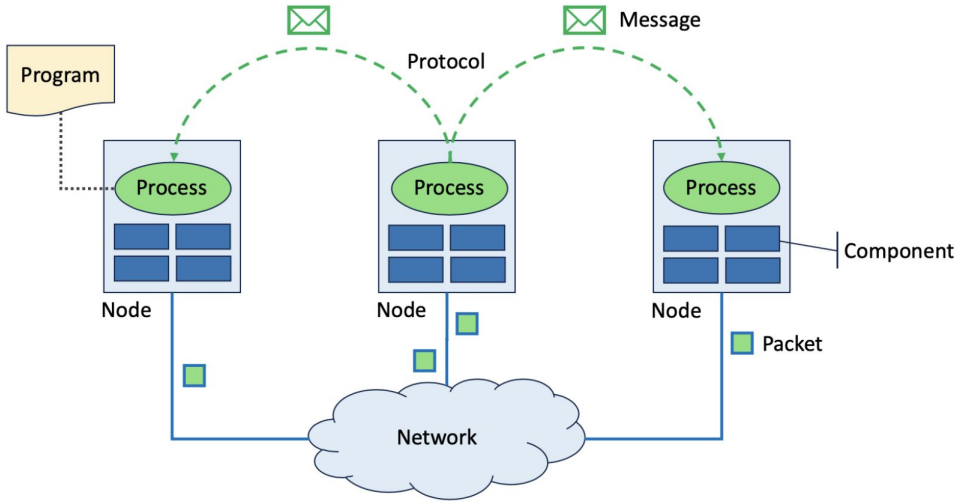
Стартовые значения

- ResNet-50 с minibatch size = 256 images обучается 29 часов на 8 Tesla P100 GPU

- Спойлер: научимся обучать на 256 видеокартах за **1 час** без потери качества

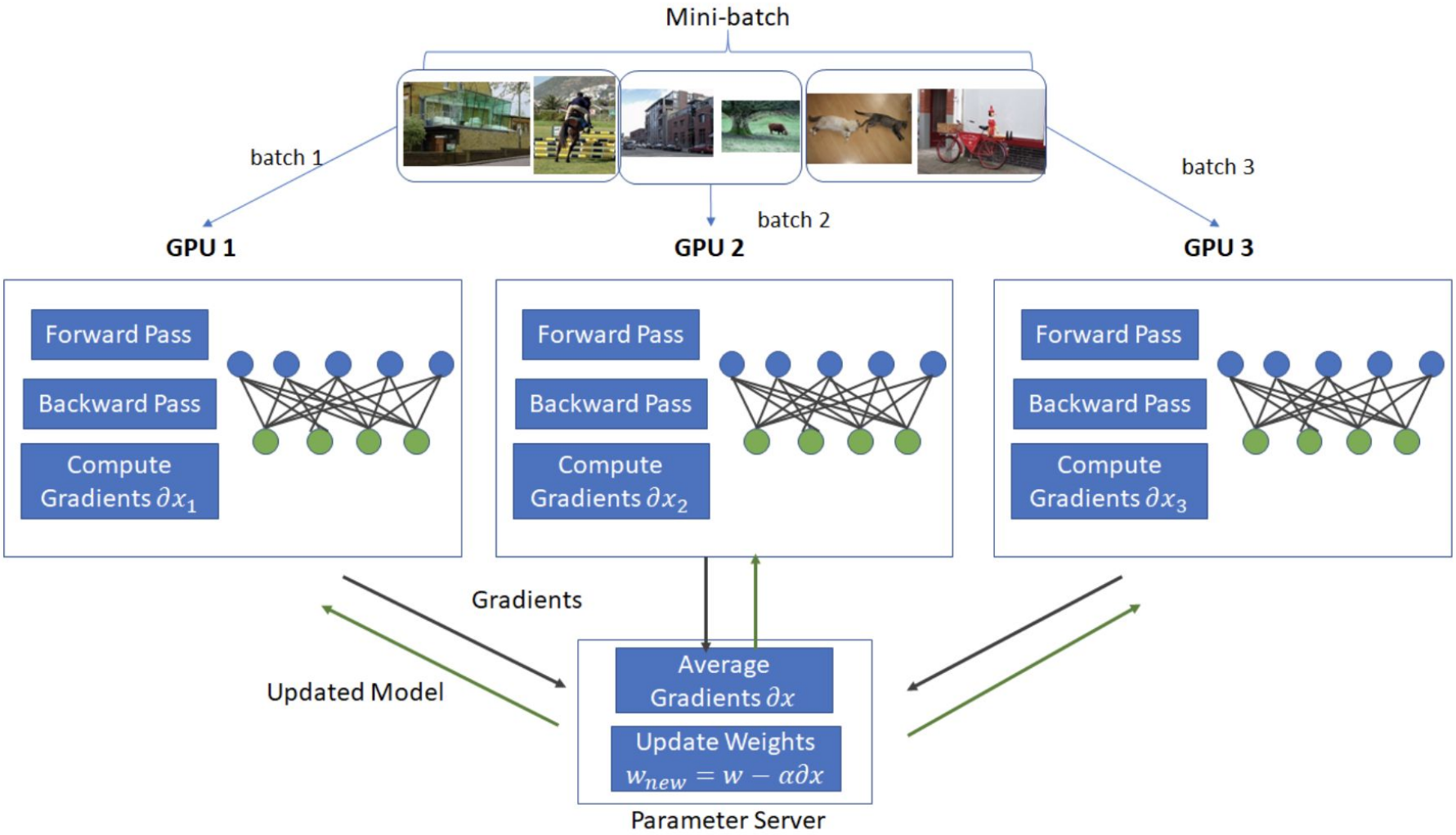
А если у нас >1 видеокарты?

- Нода – машина с видеокартой
- Пусть каждая видеокарта установлена на своей ноде
- Хотим ускорить процесс обучения



А давайте попробуем параллелить нагрузку?

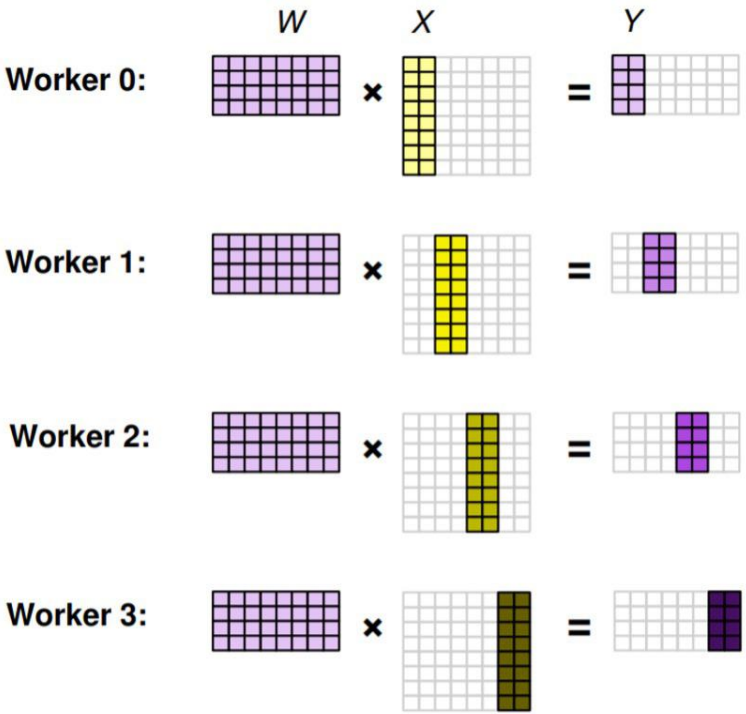
Самый простой подход – разделить batch на небольшие куски, каждый кусок дать своей ноде, а потом посчитать среднее от градиентов


Mini-batch
batch 1
batch 2
batch 3
GPU 1
GPU 2
GPU 3
Forward Pass
Backward Pass
Compute Gradients ∂x_1
Forward Pass
Backward Pass
Compute Gradients ∂x_2
Forward Pass
Backward Pass
Compute Gradients ∂x_3
Gradients
Updated Model
Average Gradients ∂x
Update Weights $w_{new} = w - \alpha \partial x$
Parameter Server

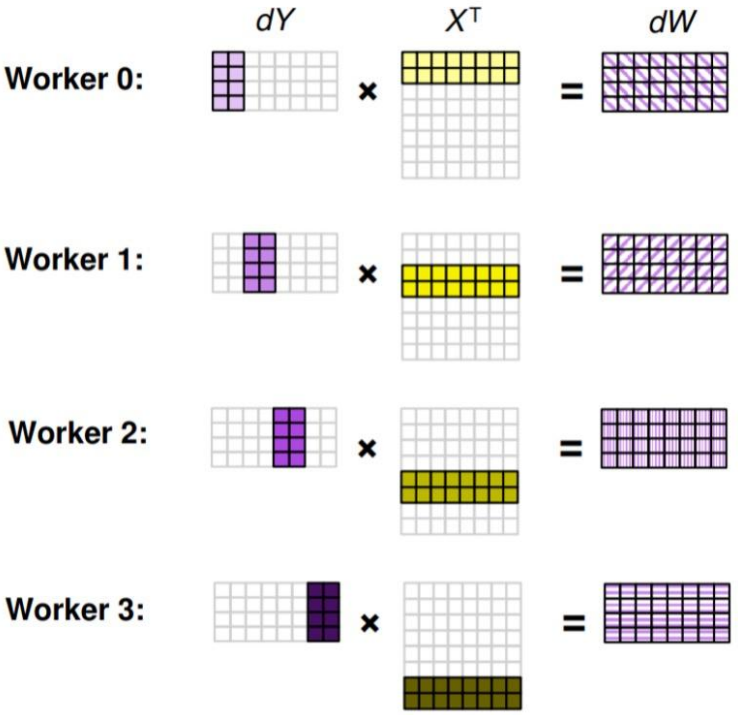
Data Parallel подход

- Каждая нода:
  - Хранит полную копию нейросети
  - Отвечает за свой кусок minibatch’а
- Forward Pass:
  - Считает значения для своего minibatch’а
  - Коммуникация между нодами не нужна
- Backward Pass:
  - Считает градиенты для своего куска
  - Считает вклад в общий градиент, основываясь на своем куске
- Обновление весов:
  - Каждая нода обновляет веса
  - Есть сервер, считающий итоговый градиенты

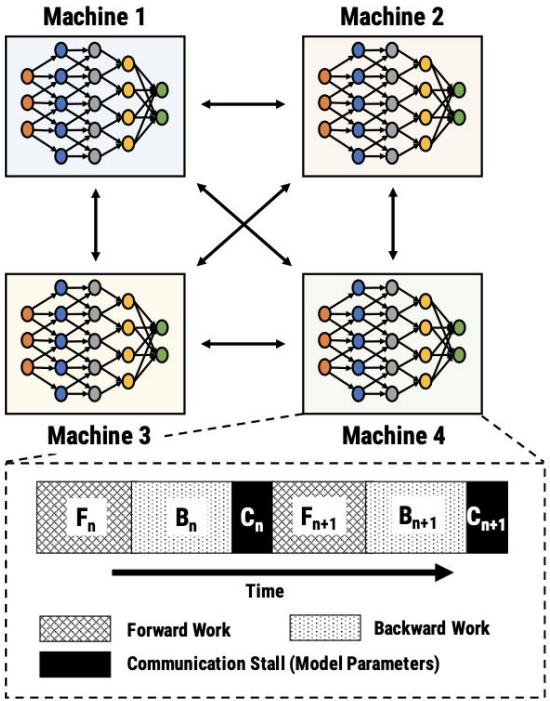
Data Parallel: Forward Pass



Data Parallel: Backward Pass



Не обязательно нужен “сервер”



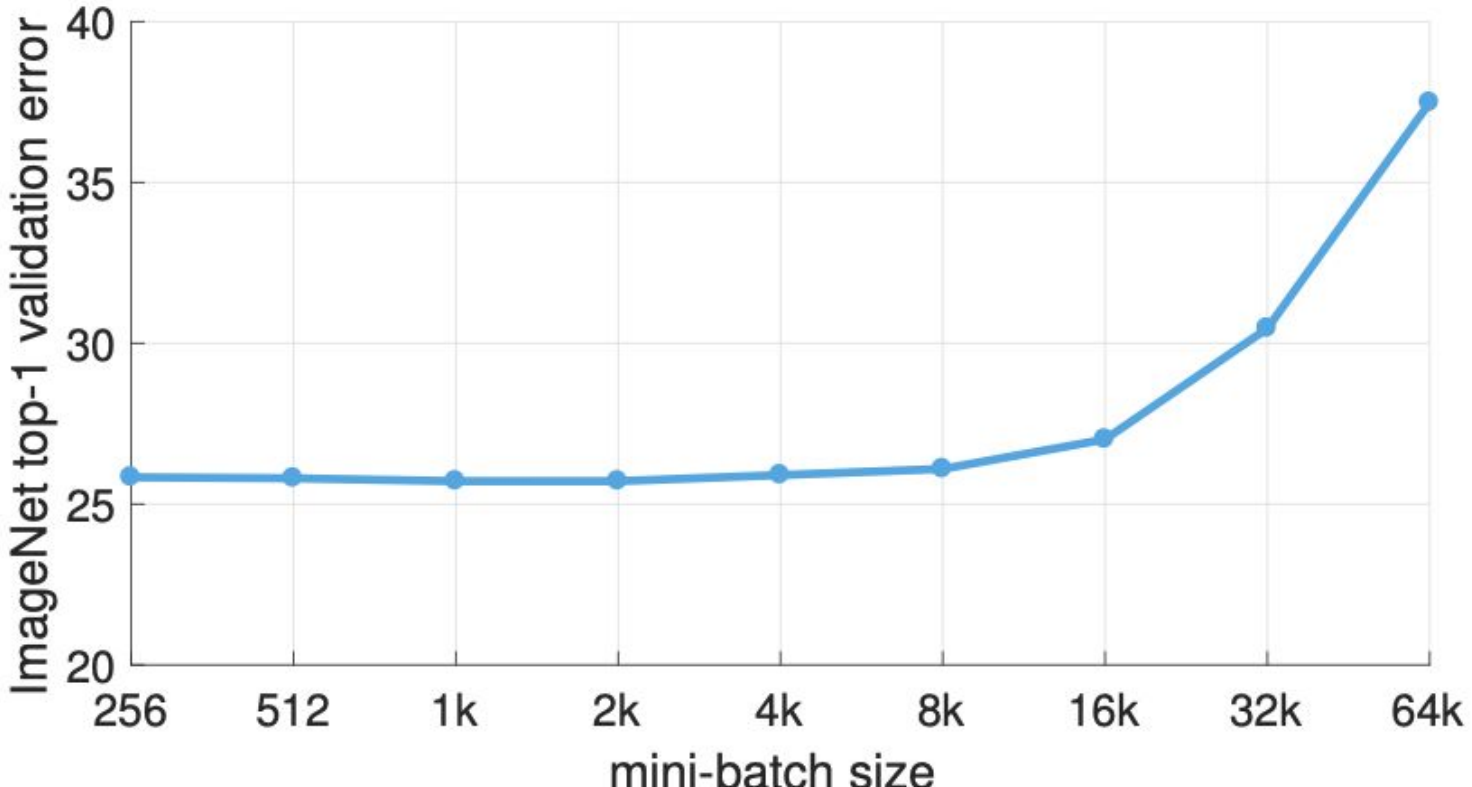
Как теперь обновлять веса?

$$\hat{w}_{t+1} = w_t - \hat{\eta}\frac{1}{k}\sum_{j<k}\nabla L(\mathcal{B}_j, w_t).$$

Где:

- *B_j* – кусок minibatch
- *L* – функция потерь
- *k* - количество нод
- *n* – learning rate, деленный на *k*

Эксперименты

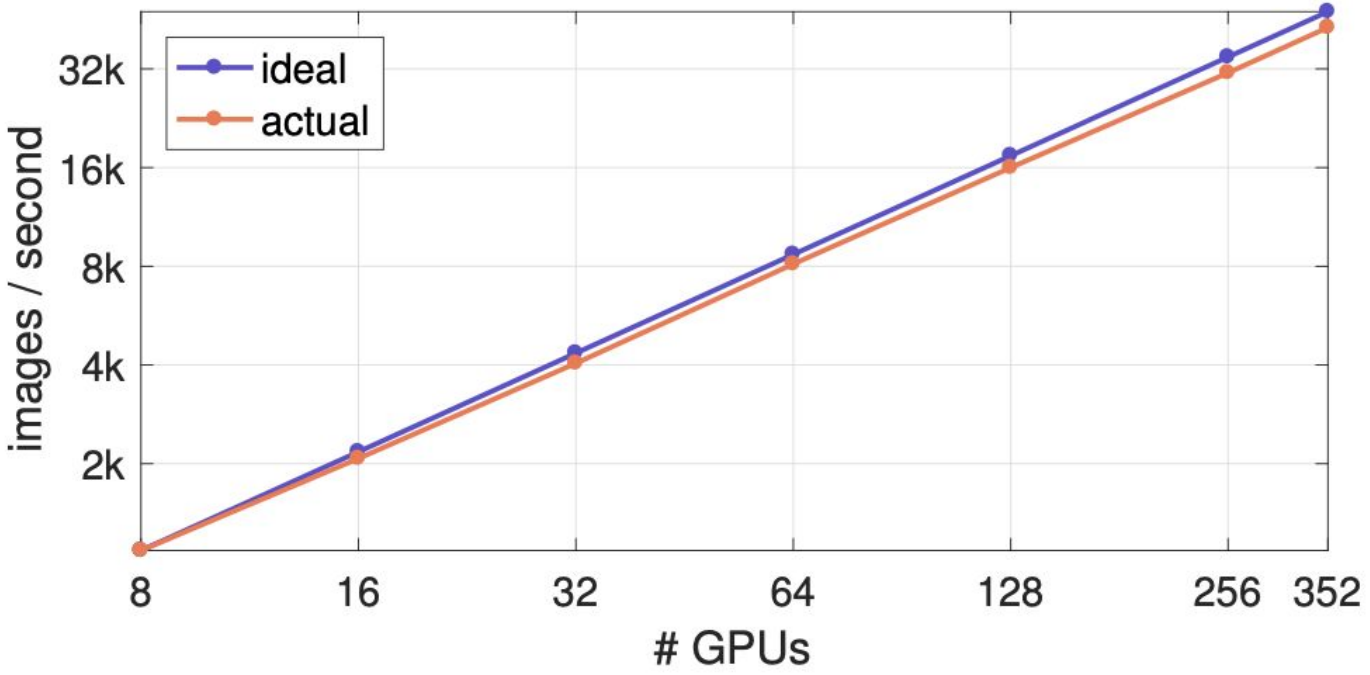


Почему падает при batch size > 16k?

- **Шум в градиентах**, застреваем в локальных градиентах. При маленьких batch-size шум помогает найти лучший минимум функции потерь

- **Генерализация**: модель слишком хорошо оптимизируется под тренировочные данные, переобучается

Scaling Efficiency 90%!



Data Parallel: выводы

- Смогли ускорить обучение **в 30 раз** почти без потери качества
- Использовали в 32 раза больше GPU (256) с эффективностью 90%
- Принципиально ускоряющий метод в DL

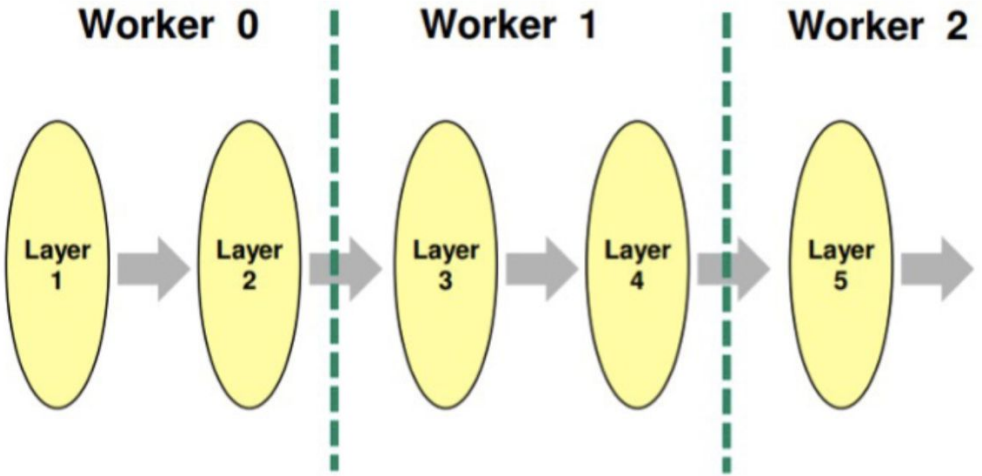
Model-parallel подход

- У нас теперь другая задача: пусть есть **очень** большая нейросеть, которая не может поместить все свои данные в память

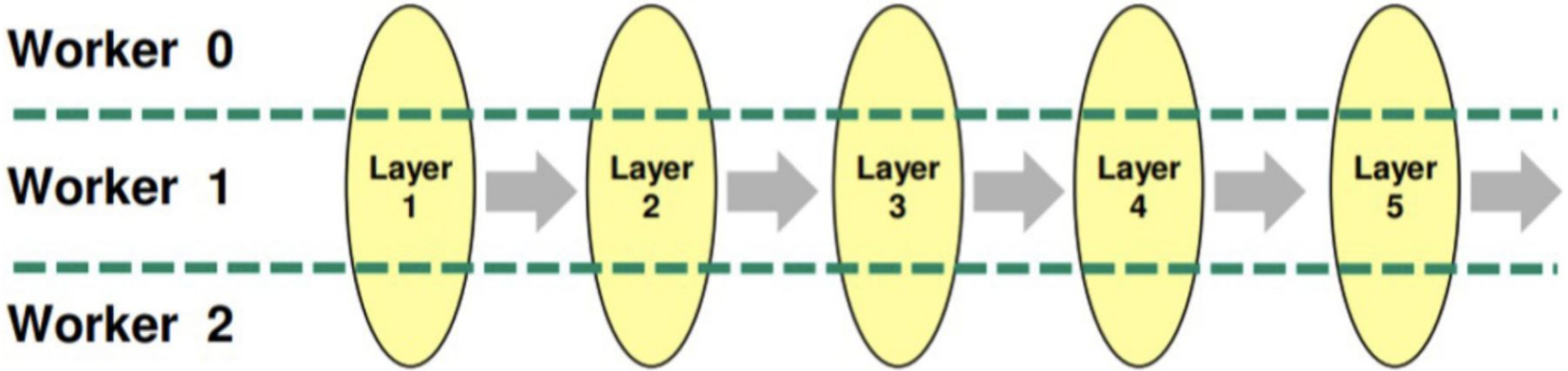
- Примитивный подход: подсчитываем несколько слоев, перекладываем данные в медленную память, загружаем новые слои и так далее.

Model-parallel подход (Inter-Layer)

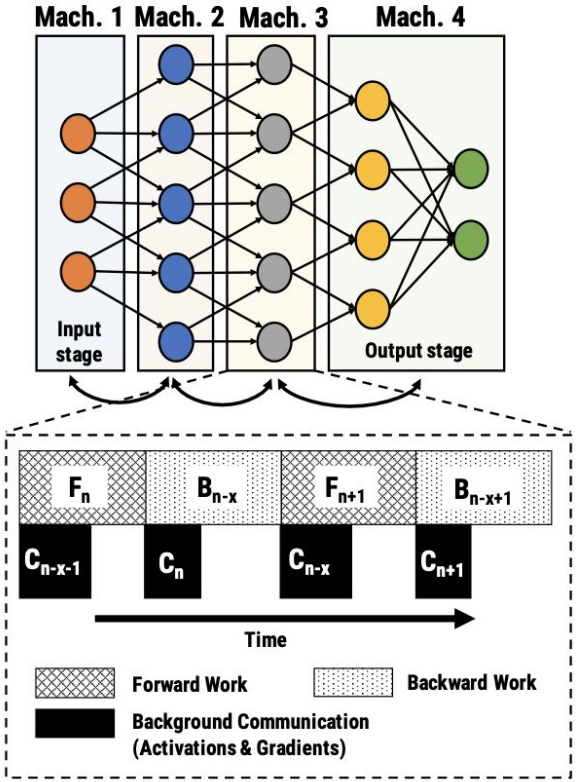
А давайте разделим слои по нодам? Каждый будет считать свои слои и передавать дальше



Model-Parallel (Intra-Layer)



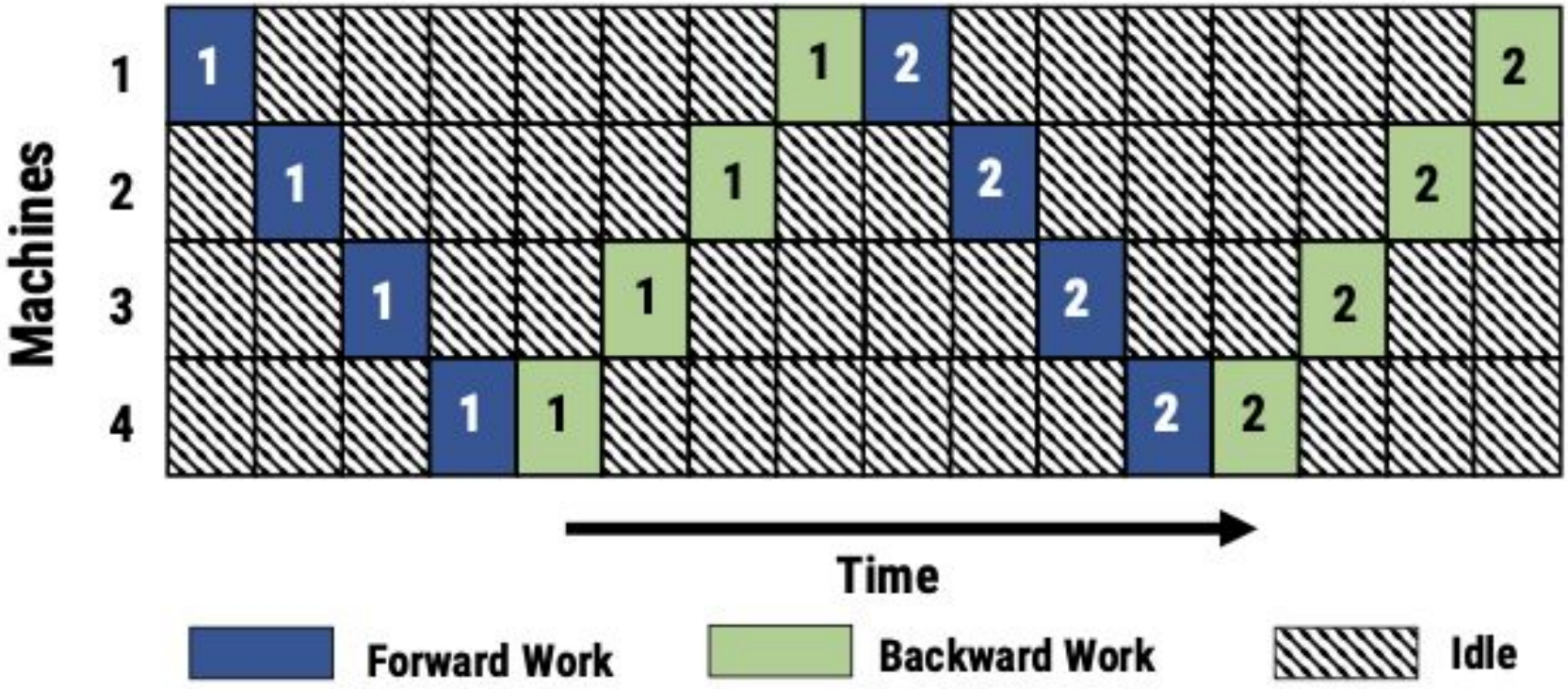
Model-parallel подход



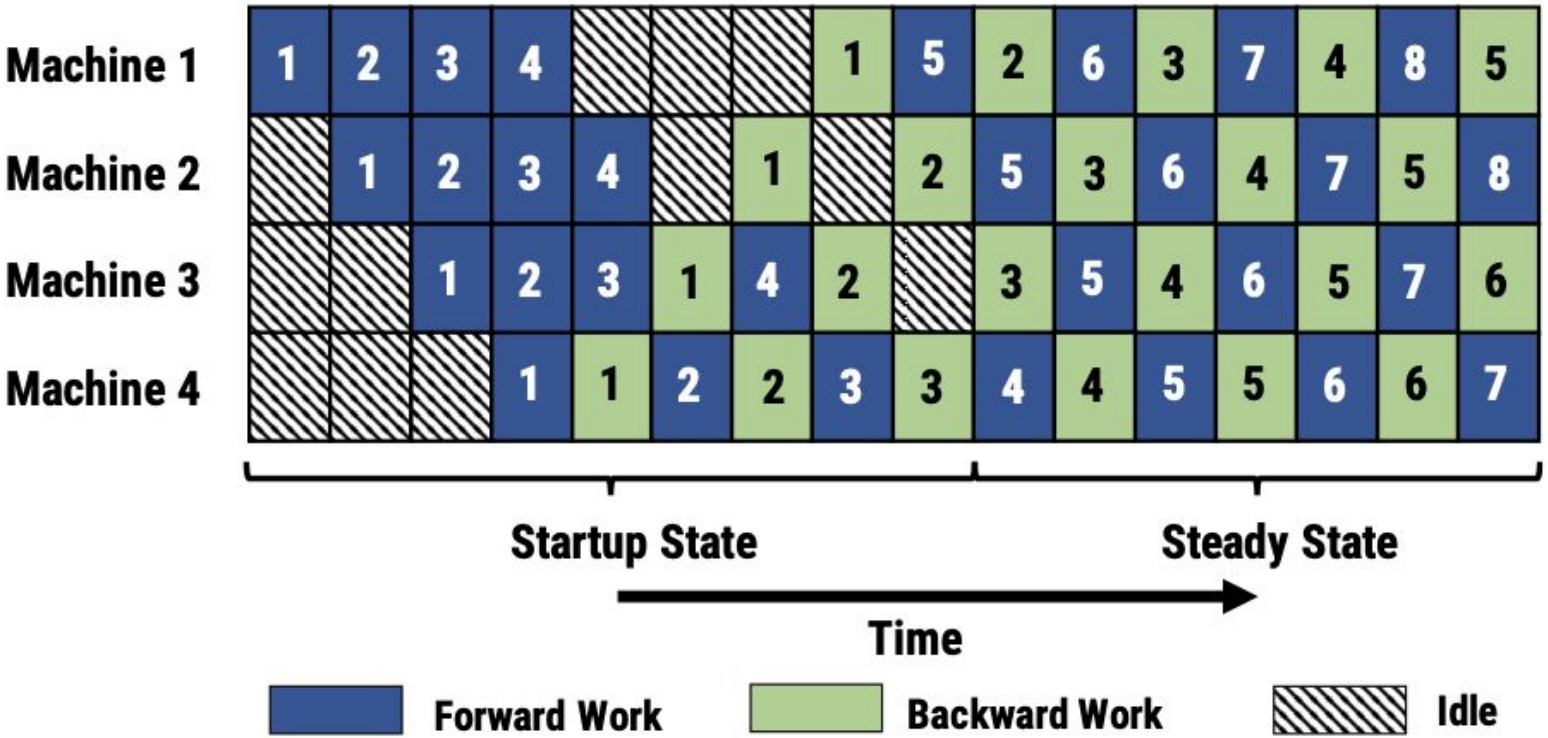
Стартовые значения

- Датасет Large Scale Visual Recognition Challenge 2012
- Модель ResNet-50

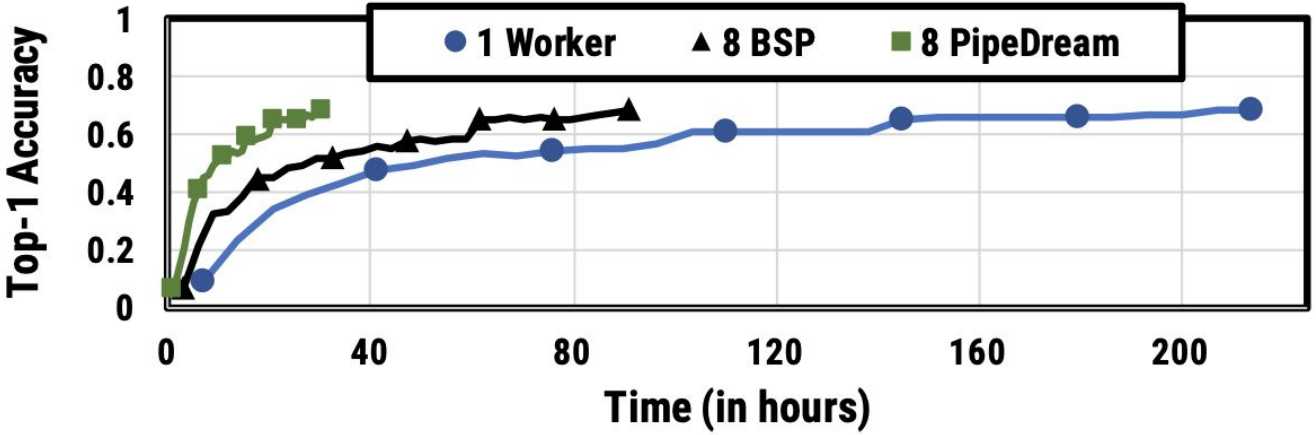
Примитивный подход



PipeLine без простоев



Сравнение времени обучения



(a) VGG16

BSP – Data Parallel подход

Главная проблема

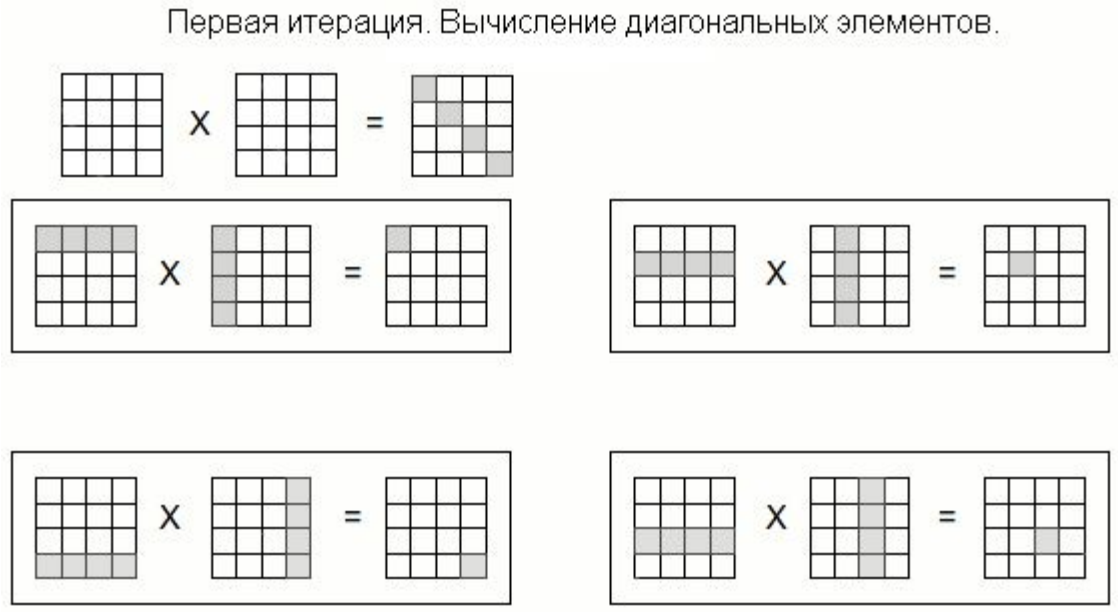
- Нужно пересылать данные между машинами. Если она одна, это достаточно быстро – но если кластер разбит по датацентрам, задержки быстро суммируются.

Tensor-Parallel подход

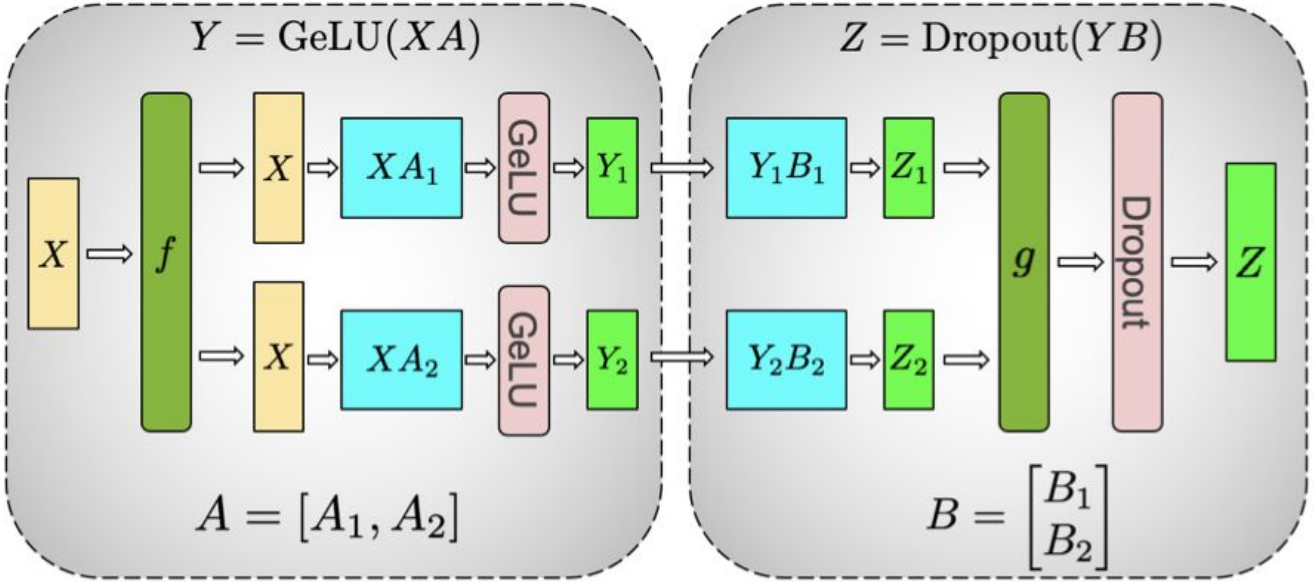
Задача все та же, но попробуем другой подход:

Будем делить не слои по разным нодам, а сами операции, которые мы можем исполнять параллельно.

Простой пример: умножение матриц

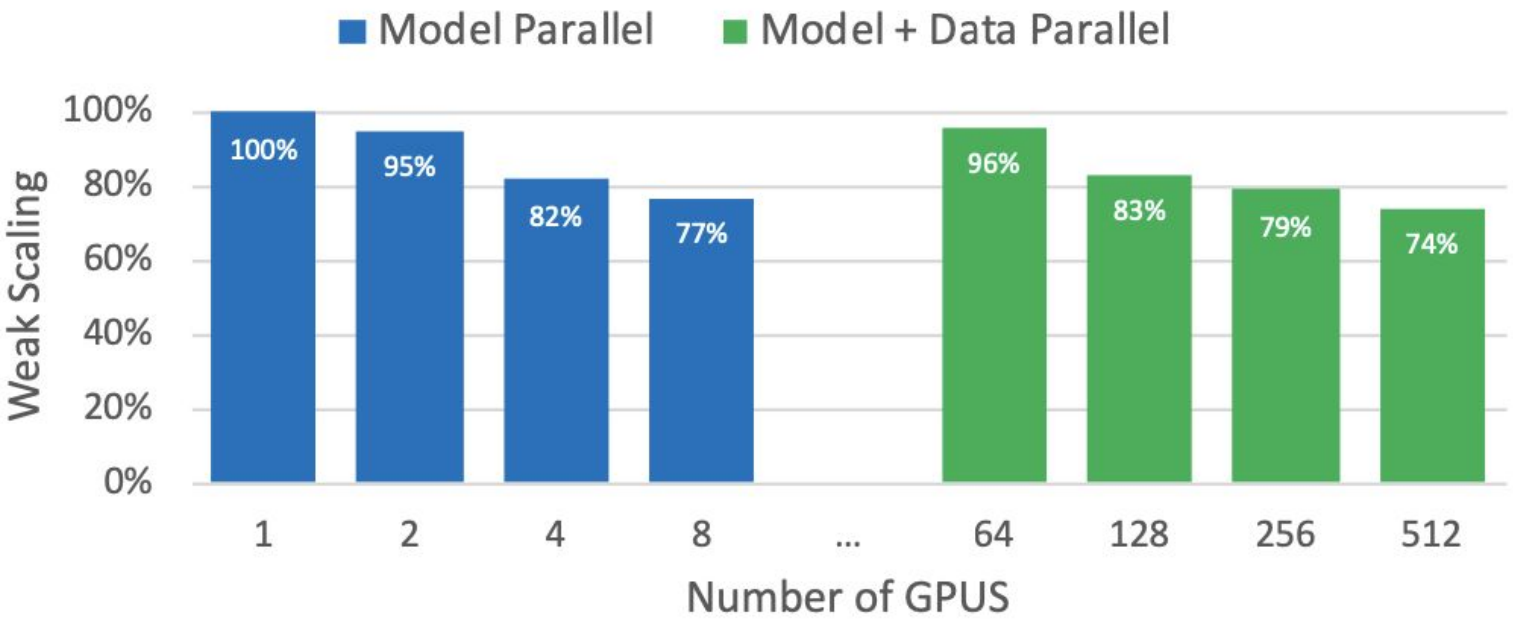


Применим в обучении нейросети



GeLU - нелинейная функция активации

Результаты экспериментов



Выводы

- С помощью Data Parallel подхода мы можем ускорять обучение на порядок, почти не теряя в качестве

- Без Model (Tensor) Parallel подхода не получилось бы обучать большие модели, такие как GPT-3 (то есть ChatGPT) или они обучались бы на порядок медленнее