Распределённое обучение нейросетей

и другие техники работы с большими данными

Лишуди Дмитрий
Петров Михаил
Набатова Дарья

Современные модели и наборы данных

- Современные нейросети очень большие, миллиарды параметров
  - GPT-3: 175 млрд параметров
  - 1 млрд 32-битных переменных занимают 4Гб памяти
- Чем больше параметров - тем больше требуется данных; увеличение размера датасета увеличивает точность
  - Для стабильного спуска батчи придётся брать достаточно большими
  - Из-за этого одна эпоха обучается очень долго

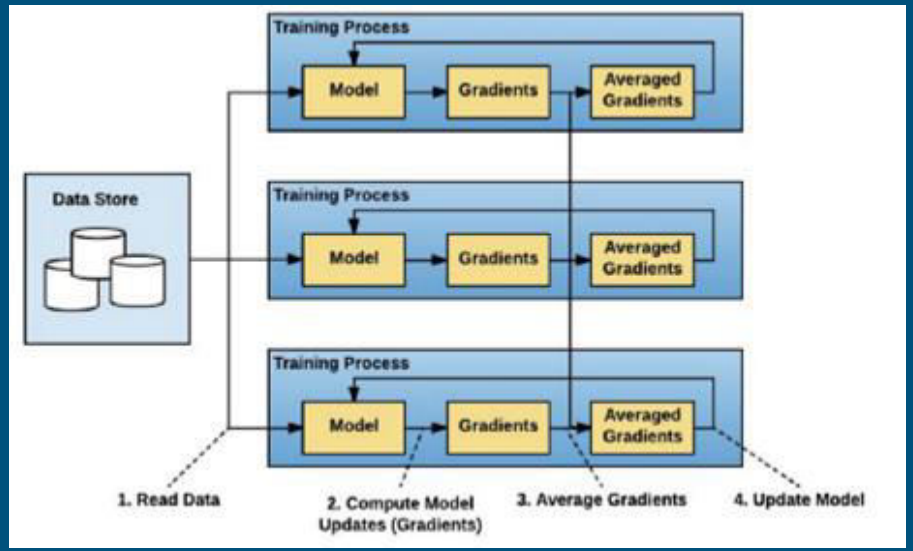
Цели

- Ускорить обучение на больших объёмах данных
- Сжать объёмы данных, занимаемых нейронной сетью
- Найти способ разделить модель между разными машинами

Какие решения бывают?

Data Parallel

- Хотим максимально задействовать в обучении несколько машин
    - Поместим копию модели на каждую машину
- Обучение должно быть эквивалентно обучению на одной машине
    - Но если очень хочется, можно это нарушить
- Паралеллим батчи
    - Машины независимо делают шаги, затем обмениваются градиентами



Фундаментальные проблемы

- Выбирая размер батча, хотим золотую середину между скоростью и точностью:
  - Маленькие батчи - нестабильное обучение
  - Слишком большие батчи - маленький прирост качества
- Не любую нейросеть получится легко распараллелить
  - BatchNorm требует усреднения по всему батчу
    - Можно усреднять по микробатчам
  - RNN требует информации о предыдущих объектах
    - Нужно придумывать модификации

Какие особенности учесть?

- Скорость коммуникаций
  - Несколько GPU? Локальный кластер? Машины разбросанные по всей планете?
  - Стоит учитывать пропускную способность сети: лучше нагружать равномерно
- Количество
  - Машин много или мало?
- Надёжность
  - Сколько сбоев и помех ожидать в машинах? В сети?

Способы обмена градиентами

По топологии

1. Централизовано
2. Децентрализовано

По времени

1. Синхронно
2. Асинхронно

Способы обмена градиентами

По топологии

1. Централизовано
2. Децентрализовано

По времени

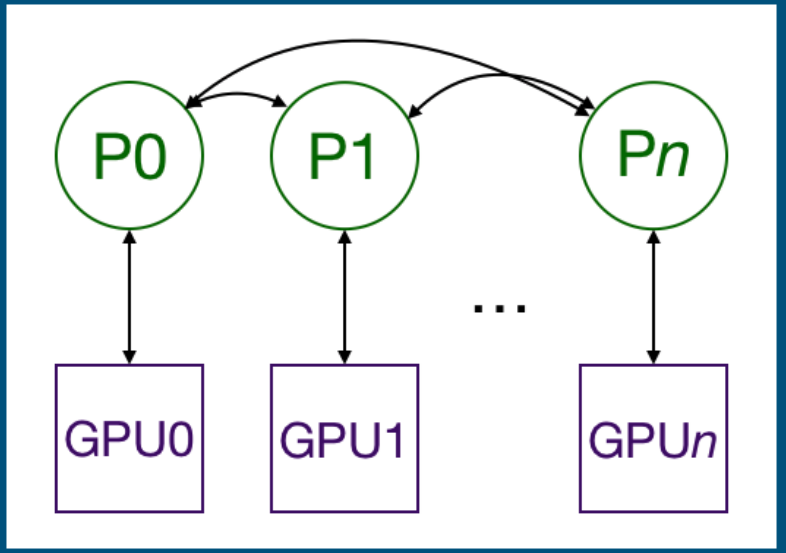
1. Синхронно
2. Асинхронно

Децентрализовано + синхронно

1. Ждём, пока все закончат прямой проход
2. С помощью AllReduce получаем средний градиент по минибатчам
3. Меняем параметры
4. Модели на всех машинах совпадают, повторяем итерацию

Децентрализовано + синхронно

1. Ждём, пока все закончат прямой проход
2. С помощью AllReduce получаем средний градиент по минибатчам
3. Меняем параметры
4. Модели на всех машинах совпадают, повторяем итерацию

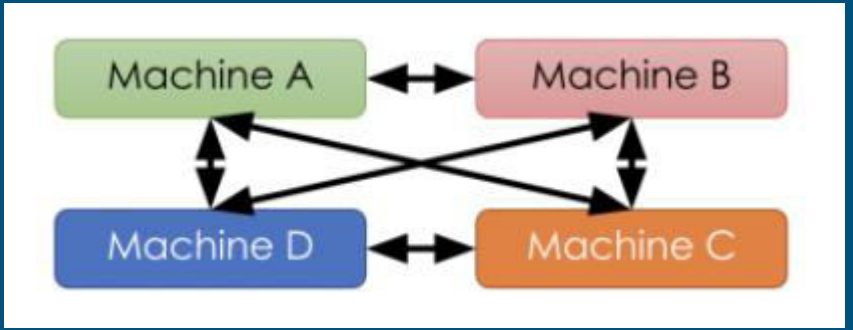


Топологии коммуникаций

- Имеем M машин.
- Каждая машина хранит P байт результатов.
- Хотим, чтобы каждая машина узнала результаты всех других

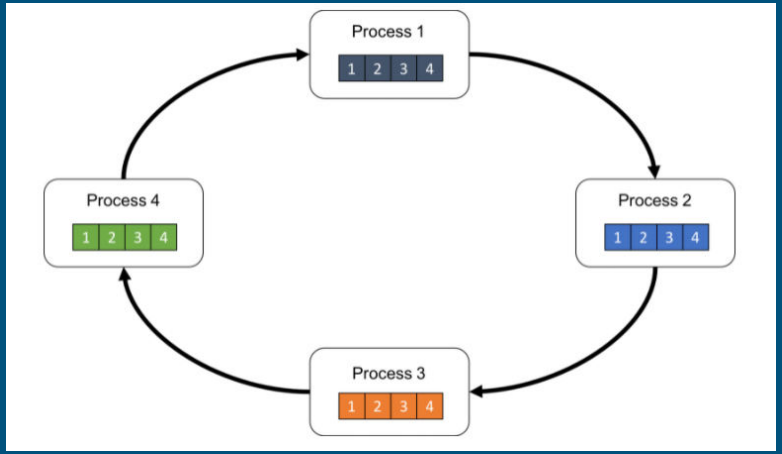
Топологии коммуникаций

- Имеем M машин.
- Каждая машина хранит P байт результатов.
- Хотим, чтобы каждая машина узнала результаты всех других
- Каждый с каждым
  - 1 шаг
  - M(M-1)P переданных байт
  - Большая нагрузка на каждую машину



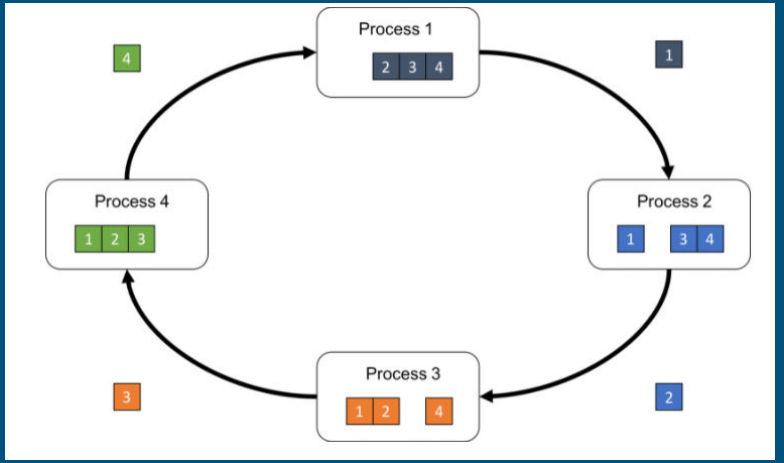
RingAllReduce

1. Делим результаты на M групп весом P/M
2. На каждом шаге получаем сумму, прибавляем свой элемент и на следующий шаг отправляем эту группу



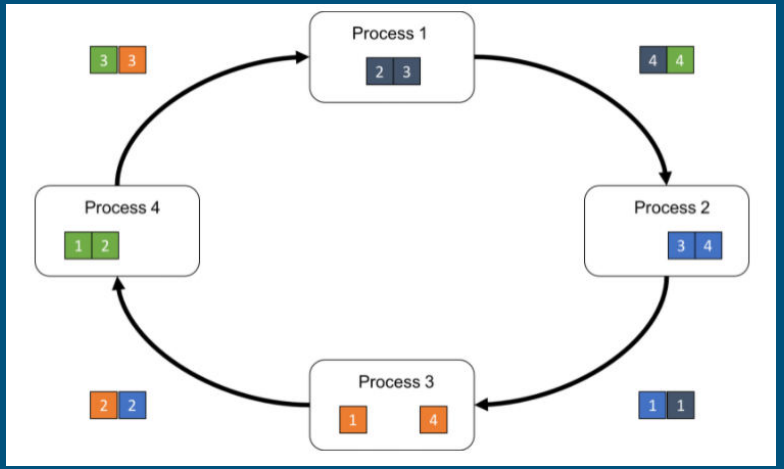
RingAllReduce

1. Делим результаты на M групп весом P/M
2. На каждом шаге получаем сумму, прибавляем свой элемент и на следующий шаг отправляем эту группу



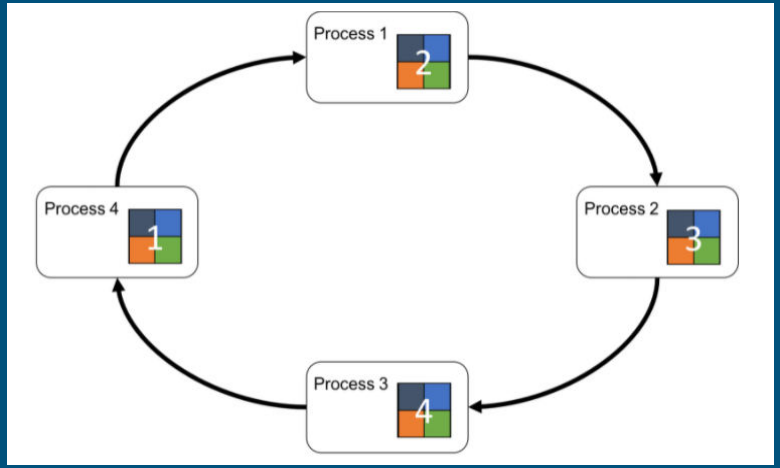
RingAllReduce

1. Делим результаты на M групп весом P/M
2. На каждом шаге получаем сумму, прибавляем свой элемент и на следующий шаг отправляем эту группу



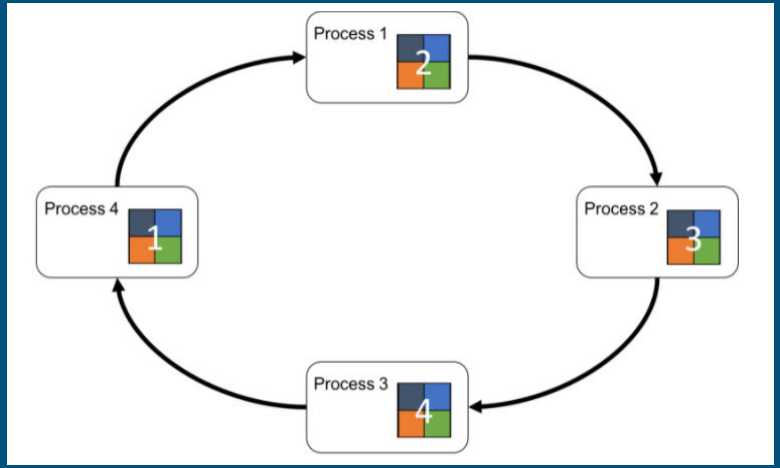
RingAllReduce

1. Делим результаты на M групп весом P/M
2. На каждом шаге получаем сумму, прибавляем свой элемент и на следующий шаг отправляем эту группу



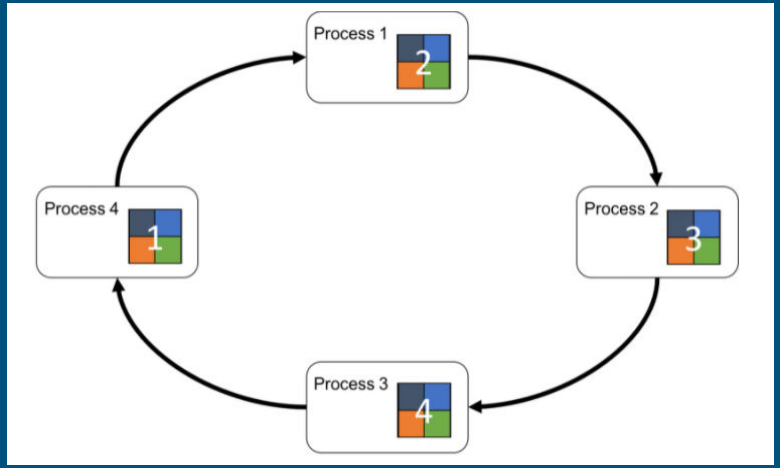
RingAllReduce

1. Делим результаты на M групп весом P/M
2. На каждом шаге получаем сумму, прибавляем свой элемент и на следующий шаг отправляем эту группу
3. После 1 круга в каждой из машин будет сумма одной группы
4. Делаем второе кольцо, в каждой машине будет сумма каждой группы



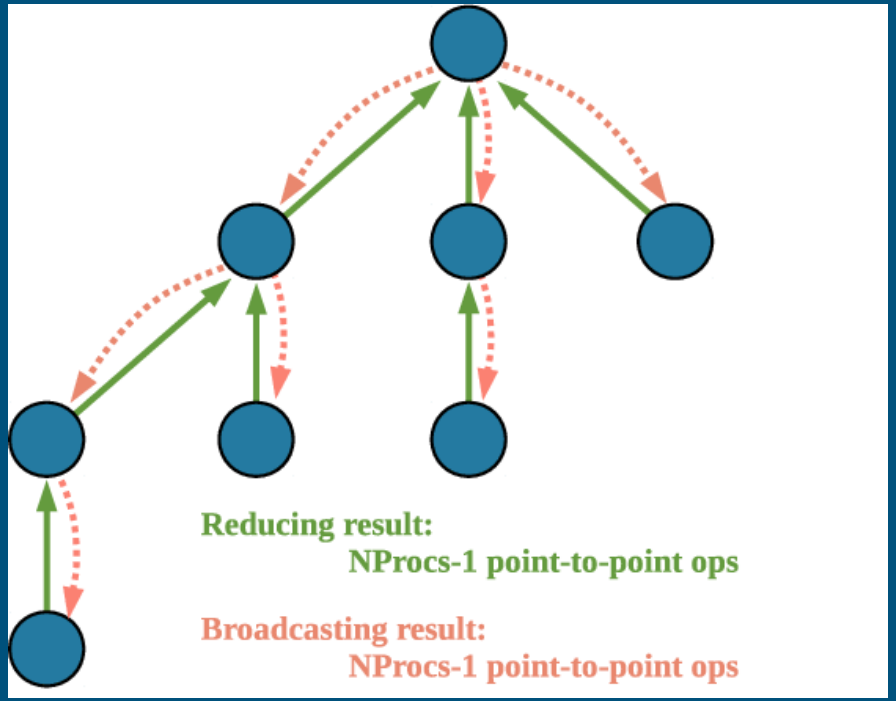
RingAllReduce

- 2(M-1) шагов
- P/M байт одна машина передаёт каждый шаг
- 2PM(M-1)/M = 2P(M-1) байт суммарно
- По 2P байт на машину - равномерно и не зависит от M!
- А если кто-то может подвести?
    - Можно нескольким машинам дать одинаковый минибатч и дождаться самую быструю
    - Нужны дополнительные рёбра и вершины



TreeAllReduce

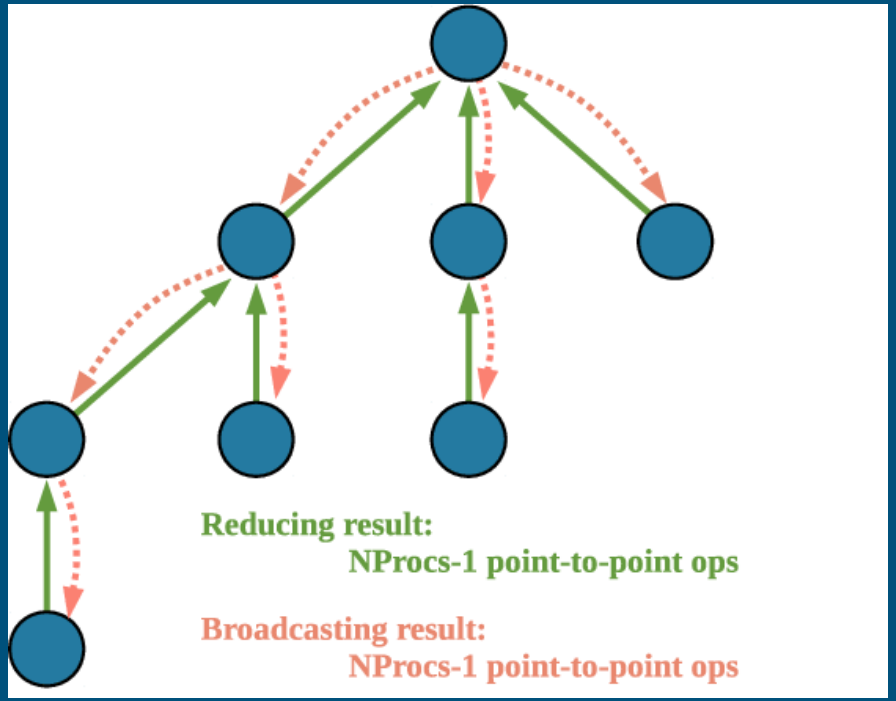
1. Передаём градиенты от листьев к корню
2. В каждом родителе суммируем
3. Передаём градиенты от корня к листьям



TreeAllReduce

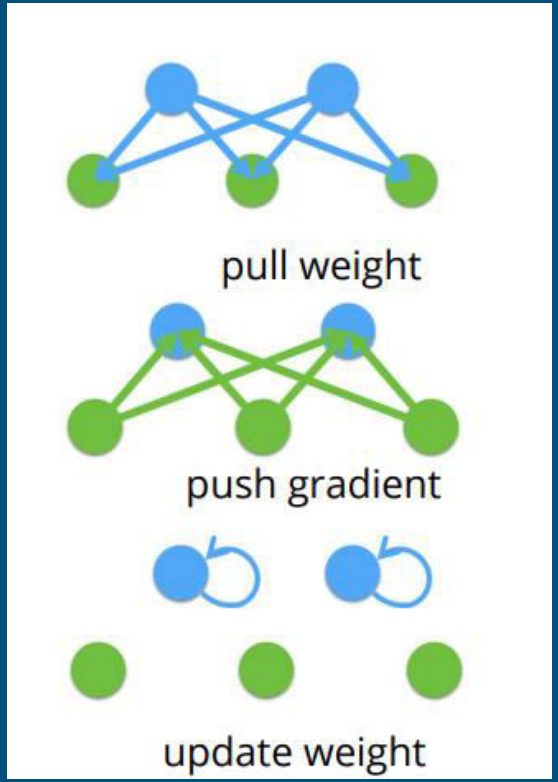
1. Передаём градиенты от листьев к корню
2. В каждом родителе суммируем
3. Передаём градиенты от корня к листьям

- Не менее 2log M шагов
- 2P(M-1) байт передать по всем машинам
- В среднем 2P, но неравномерная нагрузка
- Можно заводить лишние машины
- Можно выполнять асинхронно



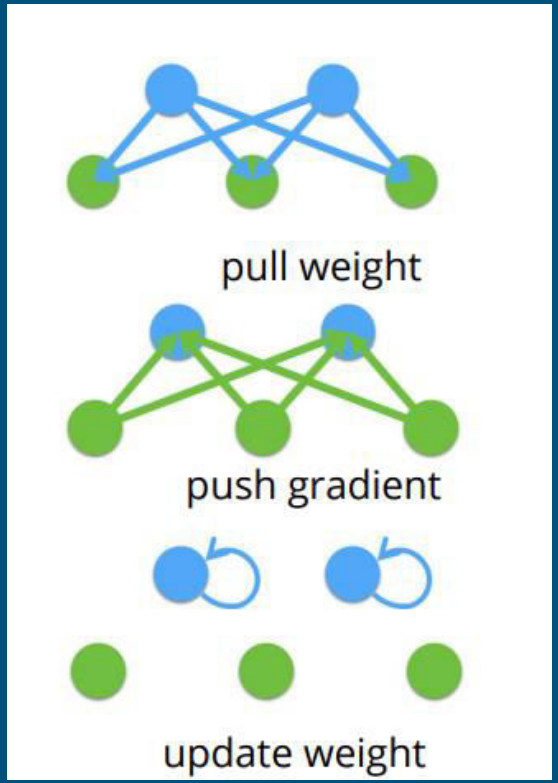
Централизовано (сервер параметров)

- Заводим отдельный серверы параметров
- Сервер собирает градиенты и редуцирует их



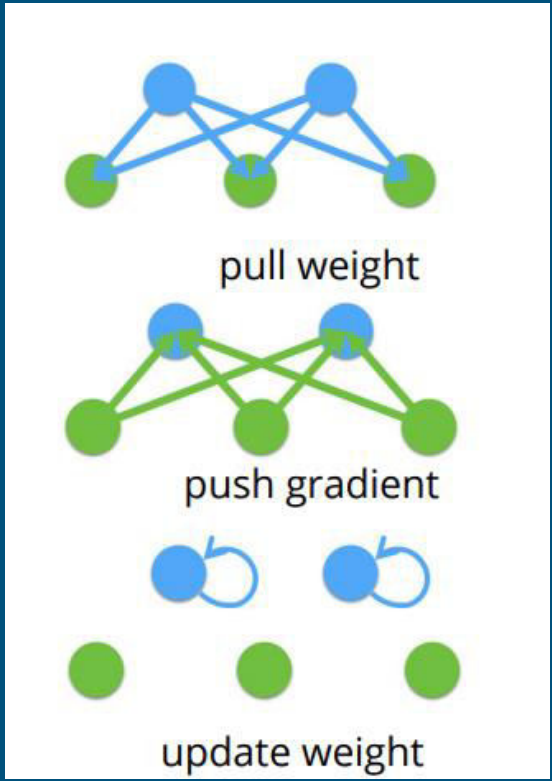
Централизовано (сервер параметров)

- Заводим отдельный сервер(ы) параметров
- Сервер собирает градиенты и редуцирует их
- Легко добавить асинхронность
  - Пока сервер редуцирует можно делать следующий шаг
  - Если градиенты каких-то машин задерживаются, можно пропустить их и усреднять по остальным



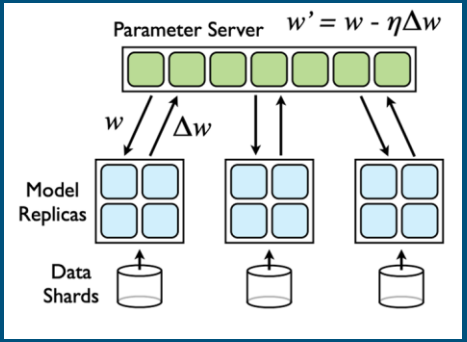
Централизовано (сервер параметров)

- Заводим отдельный сервер(ы) параметров
- Сервер собирает градиенты и редуцирует их
- Легко добавить асинхронность
  - Пока сервер редуцирует можно делать следующий шаг
  - Если градиенты каких-то машин задерживаются, можно пропустить их и усреднять по остальным
- Как синхронизировать?
  - Синхронизировать трудно, лучше использовать кольцо



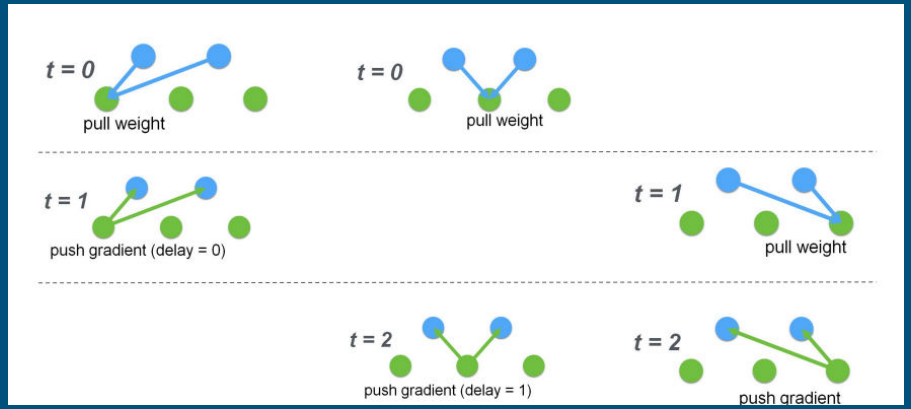
Проблемы нагрузки

- Если сервер параметров один (или их мало), у него большая нагрузка сети.
- Если серверов слишком много, то получается проблема All-to-All
- Могут быть эффективны топологии деревьев



Градиенты с задержкой

- Градиенты для асинхронного сервера получатся с задержкой
- Они показывают направление наискорейшего спуска для **прошлого набора параметров**
- Это может сильно ухудшить сходимость градиентного спуска, придётся придумывать трюки
    - Например уменьшать весь таких градиентов



Итоги

Синхронно		Асинхронно	
+	Хорошая точность; обучение в точности как на одной машине	-	Нестабильность и снижение точности из-за устаревших градиентов
-	Возможен боттлнек из-за медленной коммуникации	+	Устойчивость к задержкам сети и отказам отдельных машин

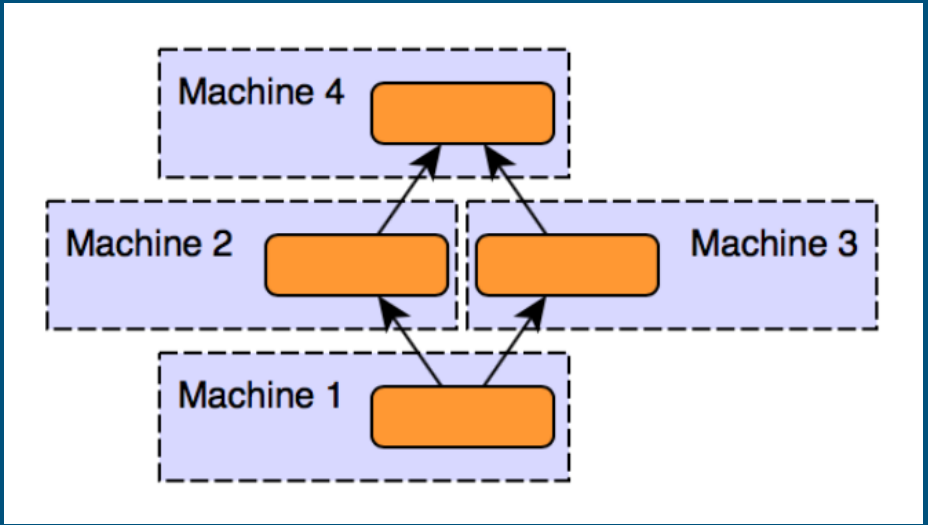
Итоги

Централизовано		Децентрализовано	
-	Большая сетевая нагрузка на один сервер, пропорциональна количеству машин. На несколько серверов требуется много коммуникаций от каждой машины	+	Можно сделать сетевую нагрузку на все машины равномерной и не зависящей от их количества с помощью кольца
-	Дополнительные траты на создание и поддержку серверов	+	Не требуется дополнительных затрат, машины обмениваются непосредственно
+	Устойчивость к задержкам сети и отказам отдельных машин	-	Задержки или сбои сети и машин сильно затормаживают решение через кольцо
+	Легко выполнять асинхронно, но синхронно выполнять не стоит	+	Легко выполнять синхронно, но можно выполнять и асинхронно

Model Parallelism

Разбиение модели на части.

- каждая машина учит свой набор параметров;
- процессы обмениваются данными при проходе вперёд-назад;
- устройства также могут отвечать за разные части данных.



Полносвязные слои

Меньше коммуникаций – лучше!

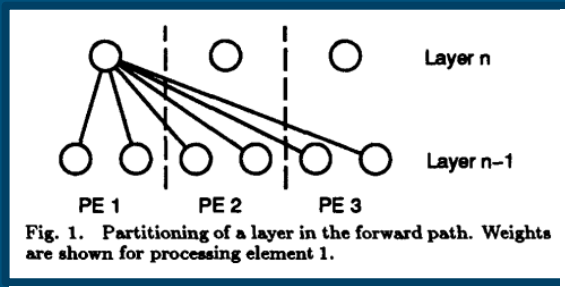


Fig. 1. Partitioning of a layer in the forward path. Weights are shown for processing element 1.

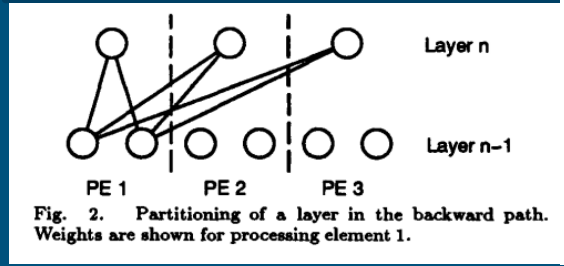


Fig. 2. Partitioning of a layer in the backward path. Weights are shown for processing element 1.

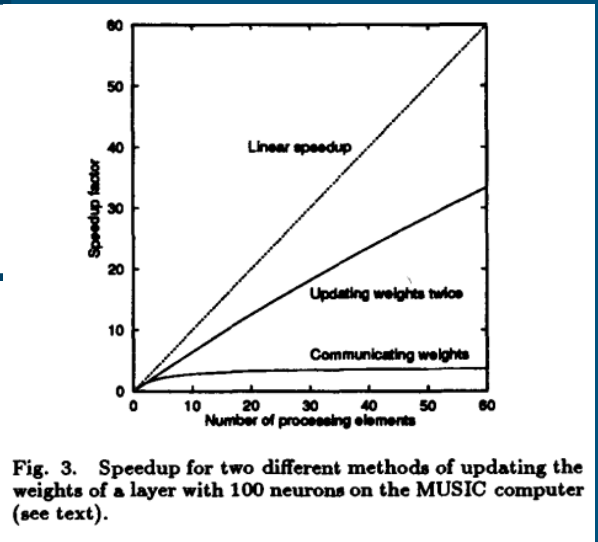
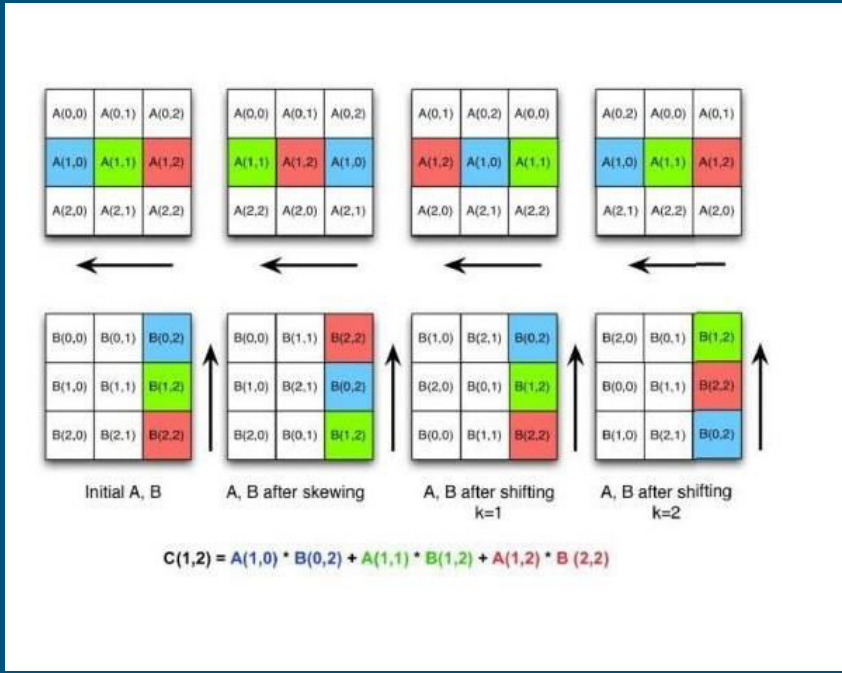


Fig. 3. Speedup for two different methods of updating the weights of a layer with 100 neurons on the MUSIC computer (see text).

Изображения: U. A. Muller and A. Gunzinger. 1994. Neural net simulation on parallel computers. In Neural Networks, IEEE International Conference on, Vol. 6. 3961–3966.

Полносвязные слои

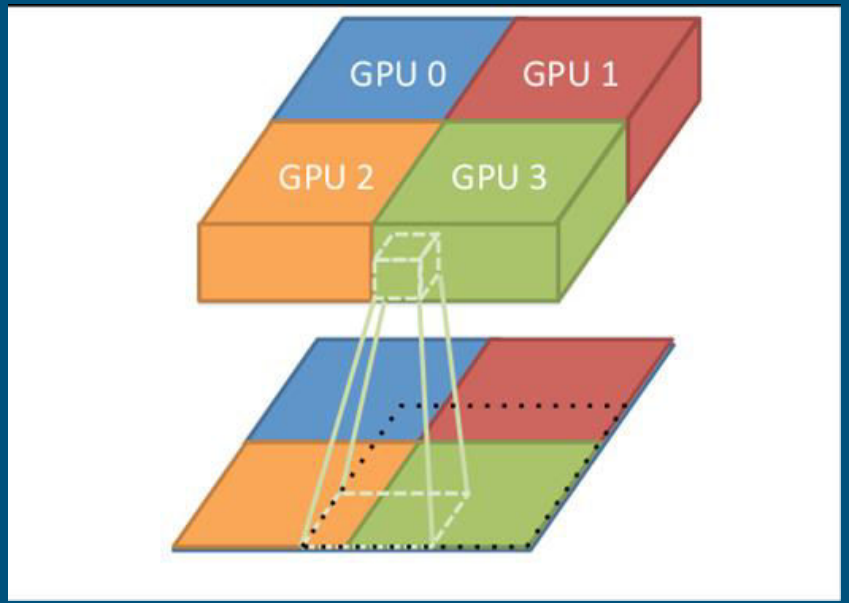
Умножать матрицы можно более или менее хитро.



$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

Свёрточные слои

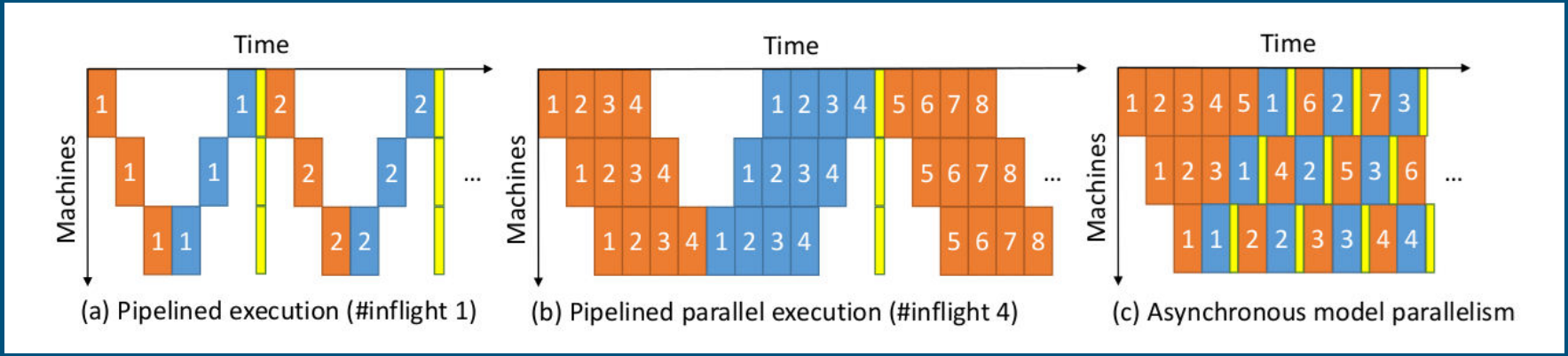
- если разбивать ядро поканально, слишком объёмная коммуникация;
- можно разбивать ожидаемый выход на области;
- обычно model parallel не применяется к свёрткам, так как по памяти проблем нет.



Изображение: A. Coates, B. Huval, T. Wang, D. J. Wu, A. Y. Ng, and B. Catanzaro. 2013. Deep Learning with COTS HPC Systems. In Proc. 30th International Conference on Machine Learning - Volume 28 (ICML'13). III–1337–III–1345.

Pipelining

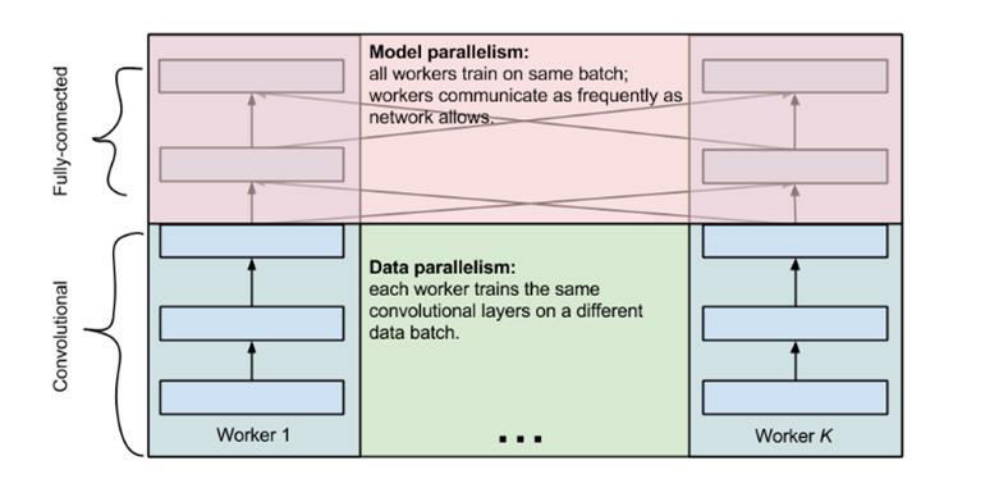
Частный случай model parallelism, когда разбиение модели имеет строго последовательную структуру.



Изображение: T. Ben-Nun and T. Hoefler. 2019. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. ACM Computing Surveys (CSUR), 52(4). 1–43.

Hybrid Parallelism

Можно комбинировать техники и использовать для каждого слоя ту технику, которая больше подходит под природу слоя.



Изображение: A. Krizhevsky. 2014. One weird trick for parallelizing convolutional neural networks. (2014). arXiv:1404.5997

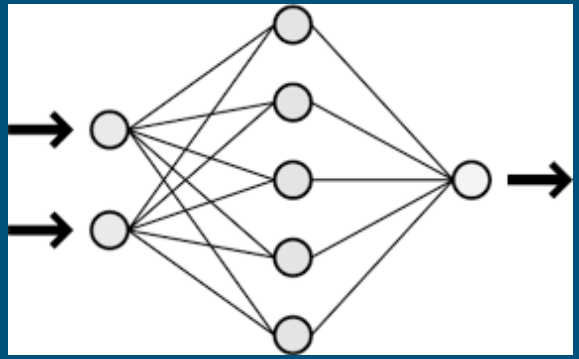
Оптимизации при работе с данными

- Работа с графами вычислений – gradient checkpoint
- Сжатие данных – mixed precision training
- Компактное представление тензоров – TT-разложение

Оптимизация при работе с графом вычислений

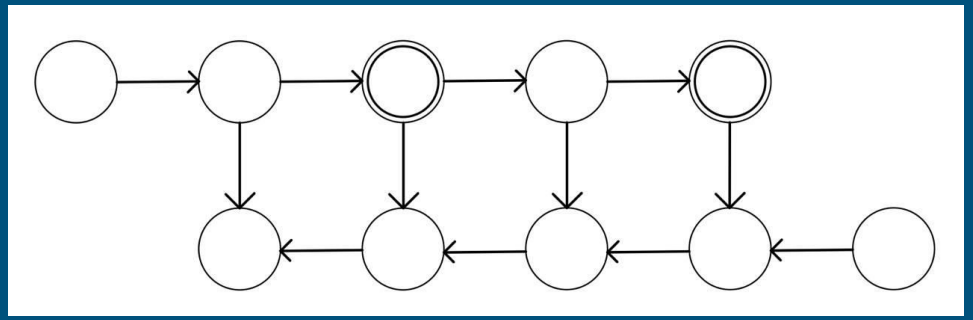
- Оптимальное решение по времени – хранить все вычисления в памяти
- Сложность по времени - $O(n)$
- Сложность по памяти - $O(n)$

- Оптимальное решение по памяти – каждый раз пересчитывать узлы
- Сложность по времени - $O(n^2)$
- Сложность по памяти - $O(1)$



Gradient checkpoint

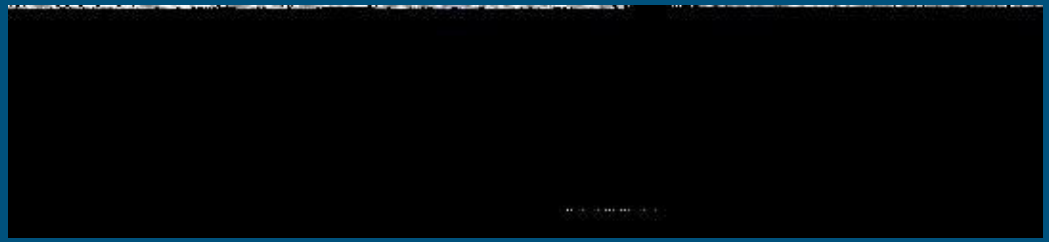
- Хотим иногда пересчитывать узлы, но не слишком часто
- Идея – пометить некоторые узлы
- Непомеченные узлы хранятся в памяти до тех пор, пока больше не потребуются



Простая нейронная сеть с прямой связью с n слоями

Gradient checkpoint

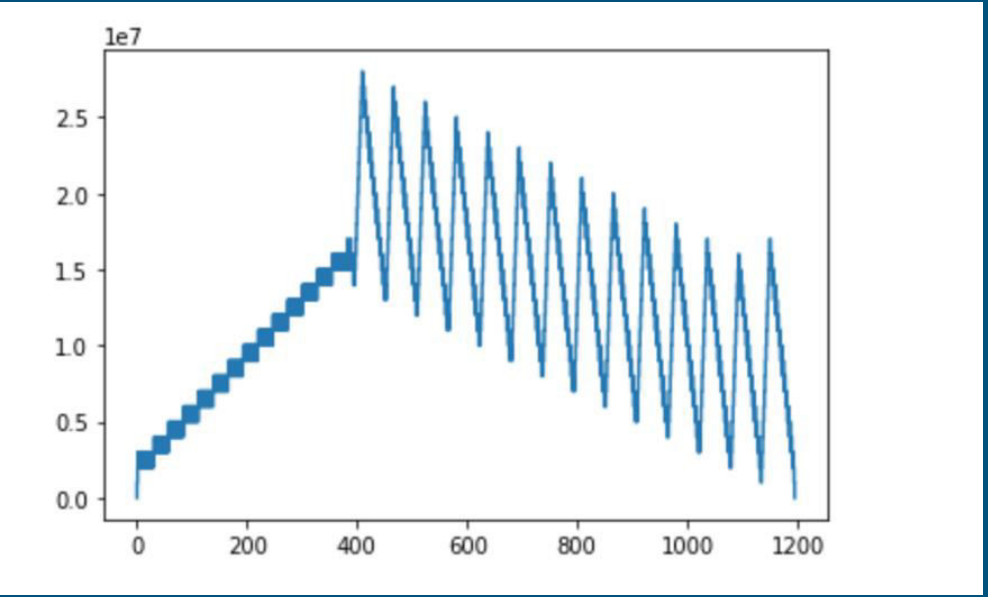
- Хотим иногда пересчитывать узлы, но не слишком часто
- Идея – пометить некоторые узлы
- Непомеченные узлы хранятся в памяти до тех пор, пока больше не потребуются



Простая нейронная сеть с прямой связью с n слоями

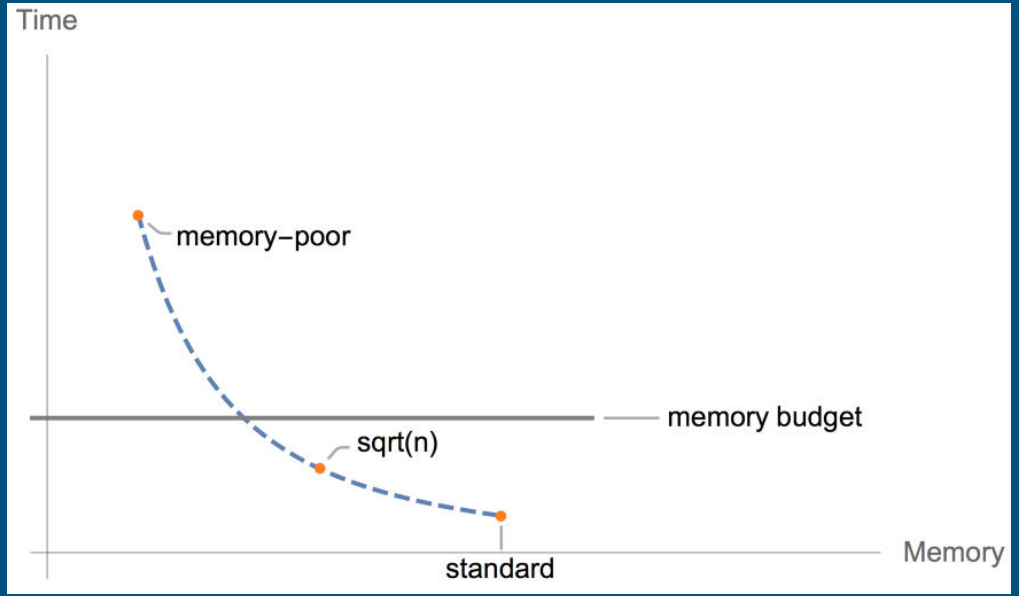
Gradient checkpoint

- Сложность по памяти - $O(\sqrt{n})$
- Сложность по времени - $O(n)$
- Проблема: нет единого алгоритма для всех графов, контрольные точки надо подбирать вручную



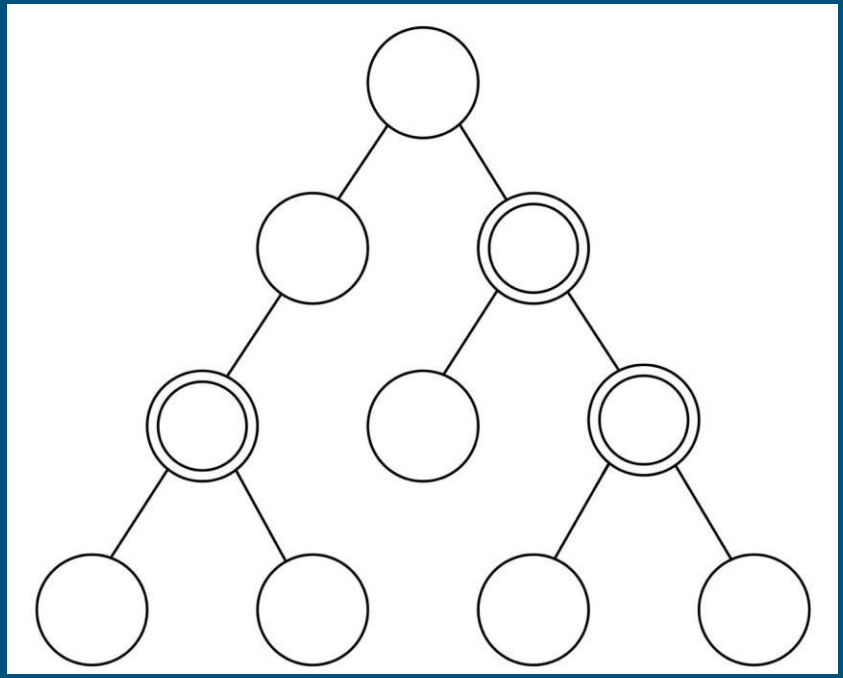
Gradient checkpoint

- Сложность по памяти - $O(\sqrt{n})$
- Сложность по времени - $O(n)$
- Проблема: нет единого алгоритма для всех графов, контрольные точки надо подбирать вручную

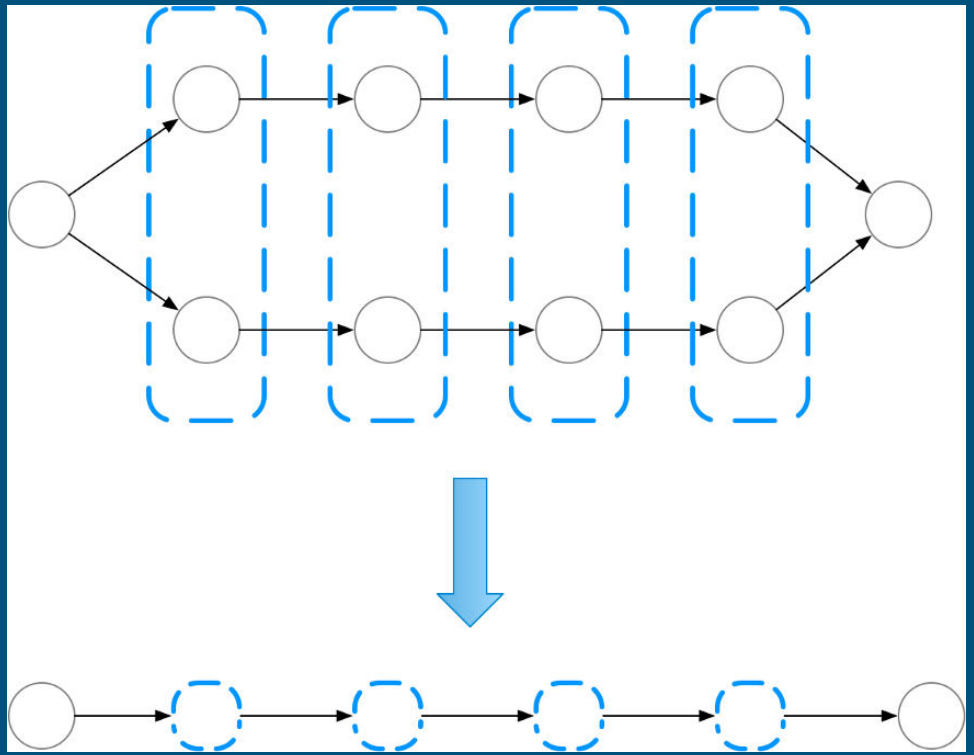


Gradient checkpoint

- Для дерева – «Разделяй и властвуй»
- Для общего графа – разложение по дереву

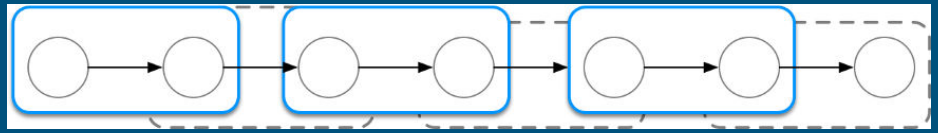


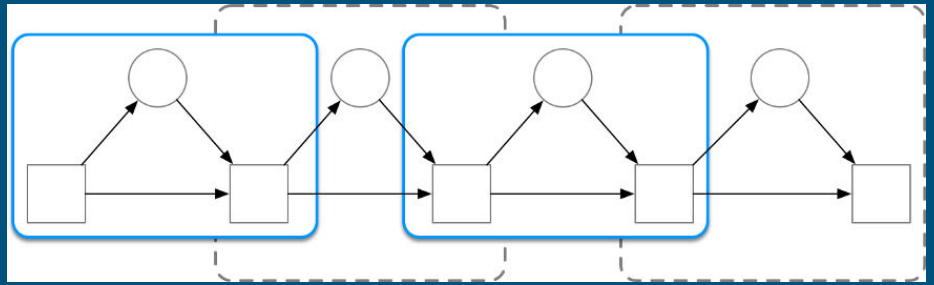
Gradient checkpoint



Gradient checkpoint

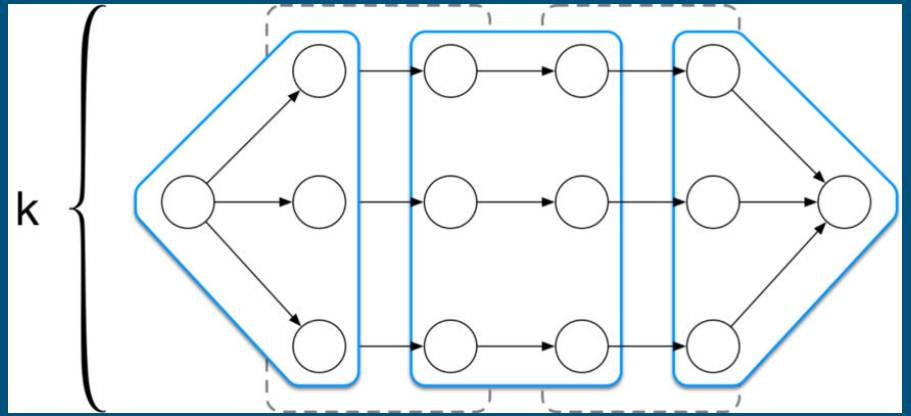
- Если граф – близок к дереву и ширина графа при этом маленькая, можем решать задачу поиска оптимального выбора чекпоинтов рекурсивно
- На практике хватает $O(n)$ времени





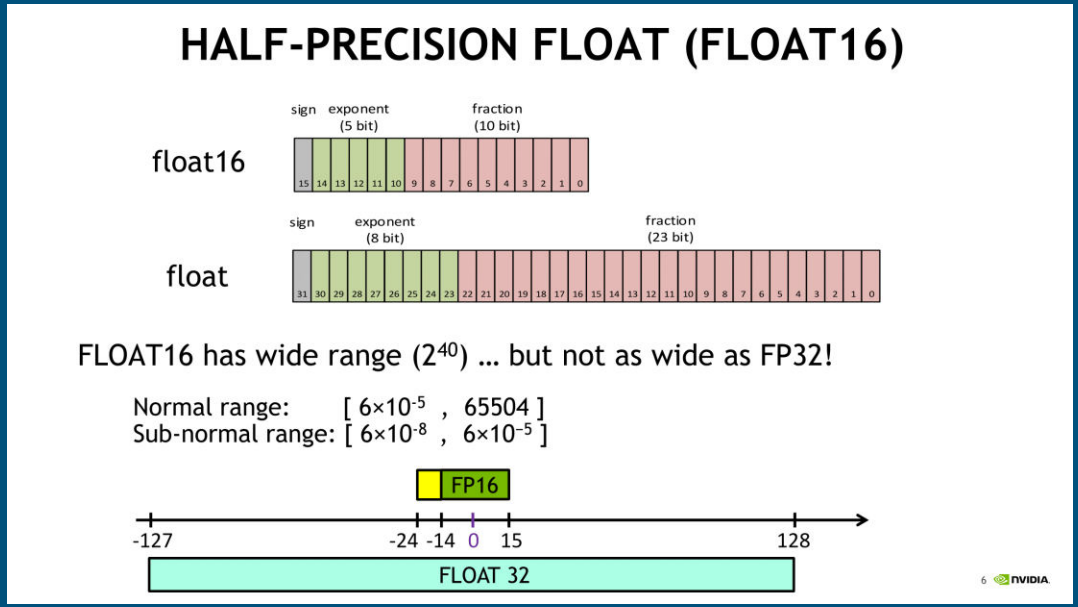
Gradient checkpoint

- Если граф – близок к дереву и ширина графа при этом маленькая, можем решать задачу поиска оптимального выбора чекпоинтов рекурсивно
- На практике хватает $O(n)$ времени



Mixed precision training

- FP16 – для хранения градиентов и весов, FP32 используется для обновления весов
- Важно масштабировать потери



Mixed precision training

- FP16 – для арифметики, хранения градиентов и копий весов, FP32 используется для хранения обновленных весов
- Важно масштабировать потери

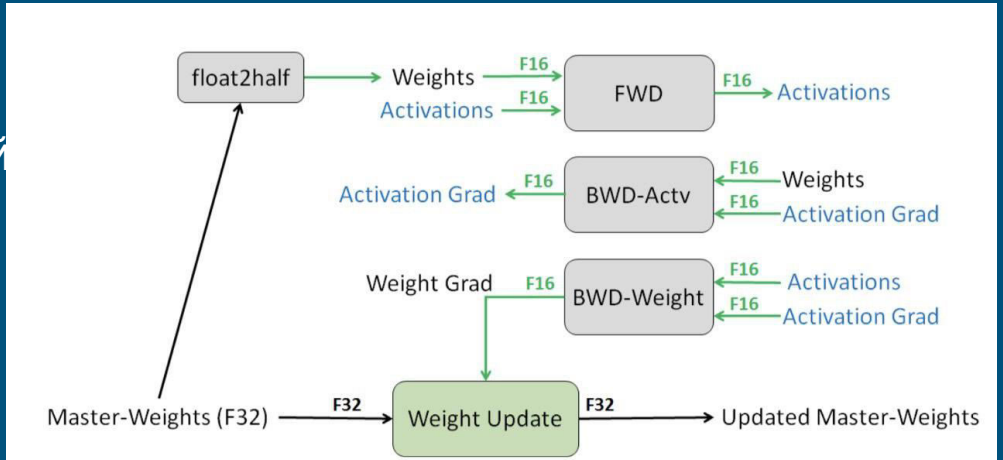
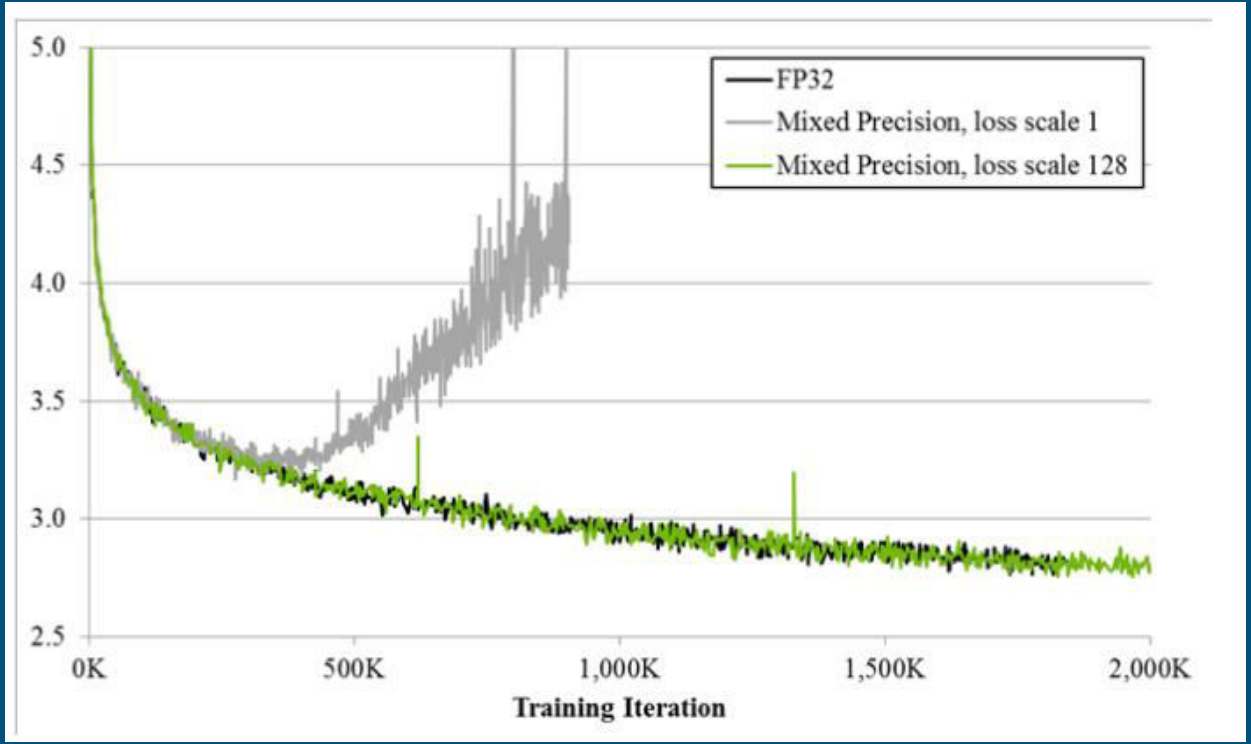


Figure 1: Mixed precision training iteration for a layer.

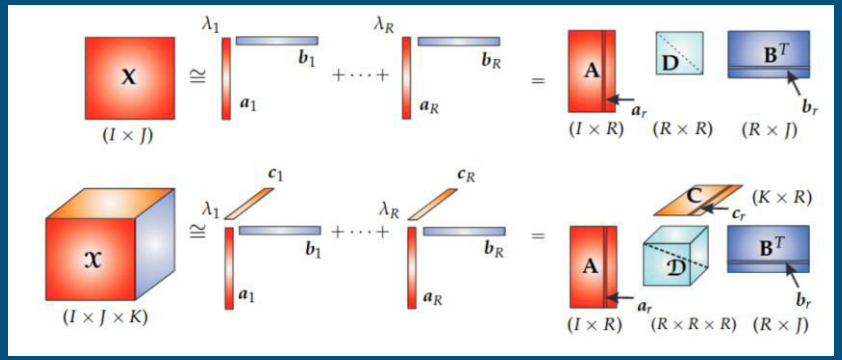
Mixed precision training



bigLSTM English language model

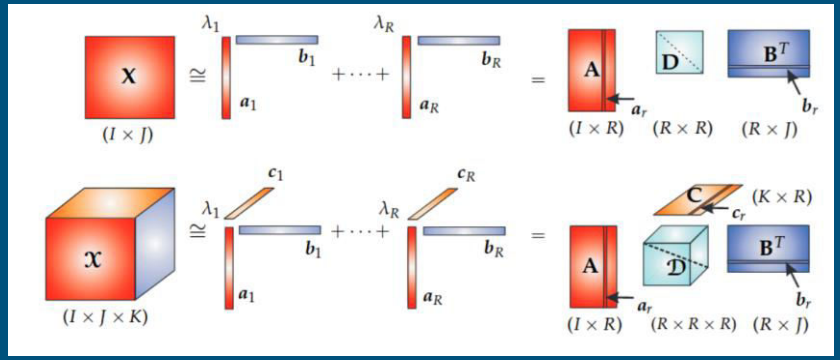
Представление тензоров

- Тензор T размерности d занимает в памяти $O(n^d)$ места
- Аналогия с SVD для матриц – SVD для тензоров. Занимает в памяти $O(rd + dnr)$ места



Представление тензоров

- SVD для матриц – представление в виде одой диагональной матрицы и двух ортогональных
- SVD для тензоров – представление в виде
  - тензора размерности d с размерами r
  - d ортогональных матриц



Представление тензоров

- Tensor Train – разложение. Занимает в памяти $O(dnr^2)$ места.
- Тензор хранится в виде d – 2 тензоров размерности 3 и двух матриц.

$$A(i_1, i_2, \ldots, i_d) = \\ = \sum_{\alpha_0, \alpha_1, \ldots, \alpha_d} G_1(\alpha_0, i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \ldots G_d(\alpha_{d-1}, i_d, \alpha_d),$$

где G_k — 3-мерные тензоры размеров $r_{k-1} \times n_k \times r_k$;
$r_0 = r_d = 1$ (вводится искусственно для удобства).

Оптимизации при работе с данными

- Работа с графами вычислений – gradient checkpoint
- Сжатие данных – mixed precision training
- Компактное представление тензоров – TT-разложение

Источники

- T. Ben-Nun and T. Hoefler. 2019. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR)*, *52*(4). 1–43. (https://arxiv.org/pdf/1802.09941).
- U. A. Muller and A. Gunzinger. 1994. Neural net simulation on parallel computers. In *Neural Networks, IEEE International Conference on, Vol. 6*. 3961–3966. (https://sci-hub.wf/10.1109/icnn.1994.374845).
- J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, and A. Y. Ng. 2010. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems 23*. 1279–1287. (https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.186.5565&rep=rep1&type=pdf).
- A. Coates, B. Huval, T. Wang, D. J. Wu, A. Y. Ng, and B. Catanzaro. 2013. Deep Learning with COTS HPC Systems. In *Proc. 30th International Conference on Machine Learning - Volume 28 (ICML'13)*. III–1337–III–1345. (http://proceedings.mlr.press/v28/coates13.pdf).
- G. Habib, and S. Qureshi. 2020. Optimization and Acceleration of Convolutional neural networks: A Survey. *Journal of King Saud University-Computer and Information Sciences*.

Источники

- https://arxiv.org/pdf/1710.03740.pdf
- https://github.com/cybertronai/gradient-checkpointing
- https://arxiv.org/pdf/1604.06174v2.pdf
- https://dlsys.cs.washington.edu/pdf/lecture11.pdf
- https://arxiv.org/pdf/1802.09941.pdf
- https://arxiv.org/pdf/2006.15704.pdf
- https://web.stanford.edu/~rezab/classes/cme323/S16/projects_reports/hedge_usmani.pdf
- http://seba1511.net/dist_blog/
- http://www.juyang.co/distributed-model-training-ii-parameter-server-and-allreduce/
- https://arxiv.org/pdf/1706.02677.pdf

Спасибо за внимание!