Распределенное обучение нейросетей

Воробьев Дмитрий
БПМИ202

Зачем учить нейросети на нескольких машинах?

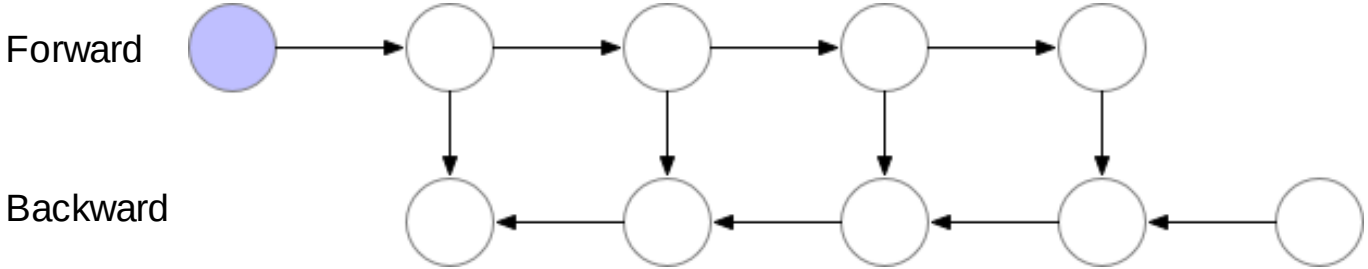
- Ускорить обучение
- Сэкономить память
- Оба пункта

Экономия памяти

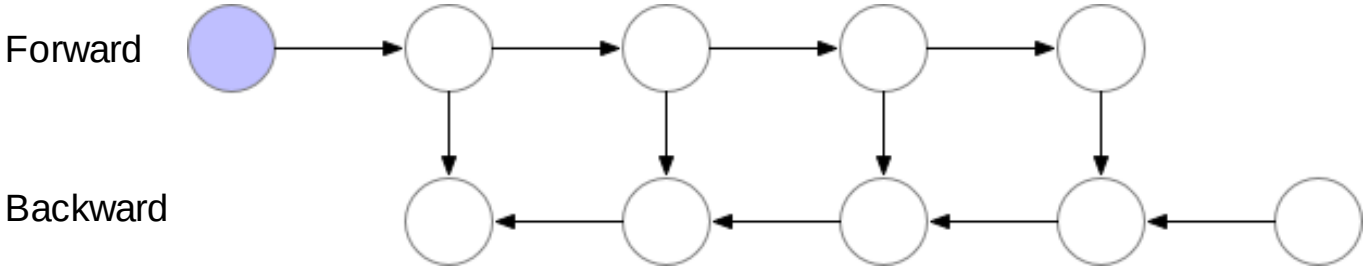
- Gradient checkpointing
- Mixed/Low precision
- Tensor/Model parallelism

Gradient checkpointing

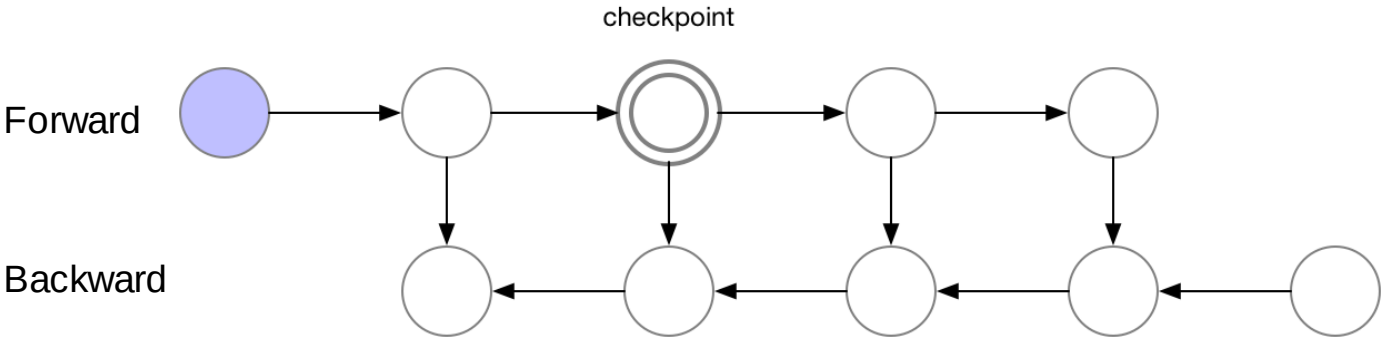
Обычный режим вычисления градиента

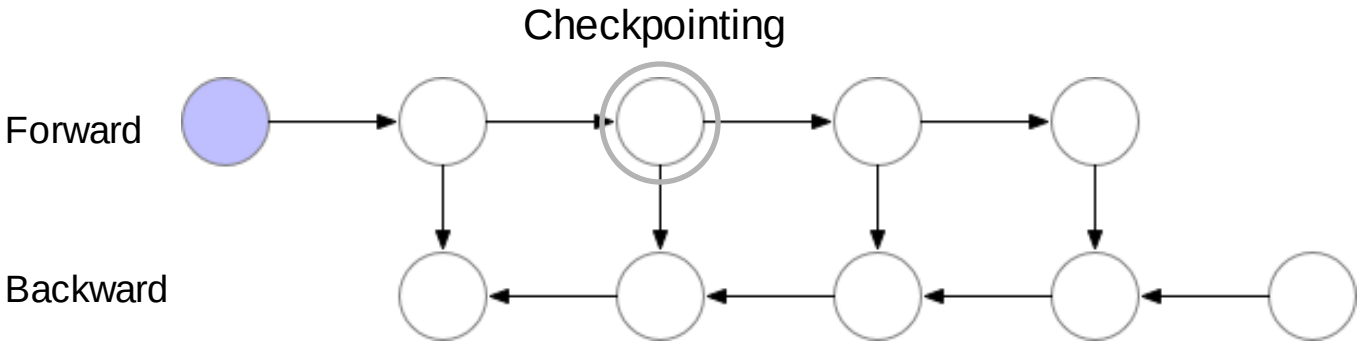


Режим с сохранением памяти



Gradient checkpointing





Mixed precision

Идея в том, чтобы использовать числа с пониженной разрядностью, где это возможно

В PyTorch есть поддержка такого функционала.

Видеокарты NVIDIA поддерживают вычисления с неравной степенью сжатия для различных данных. В PyTorch есть модуль Automatic Mixed Precision (amp), который использует алгоритмы Facebook и особенности видеокарт NVIDIA и за вас автоматически понижает точность, где это возможно.

Подробнее: https://pytorch.org/blog/accelerating-training-on-nvidia-gpus-with-pytorch-automatic-mixed-precision/

```
import torch
# Creates once at the beginning of training
scaler = torch.cuda.amp.GradScaler()

for data, label in data_iter:
   optimizer.zero_grad()
   # Casts operations to mixed precision
   with torch.cuda.amp.autocast():
      loss = model(data)

   # Scales the loss, and calls backward()
   # to create scaled gradients
   scaler.scale(loss).backward()

   # Unscales gradients and calls
   # or skips optimizer.step()
   scaler.step(optimizer)

   # Updates the scale for next iteration
   scaler.update()
```

Распределенное обучение

- Data Parallelism

  Распределяем данные по машинам

- Model Parallelism

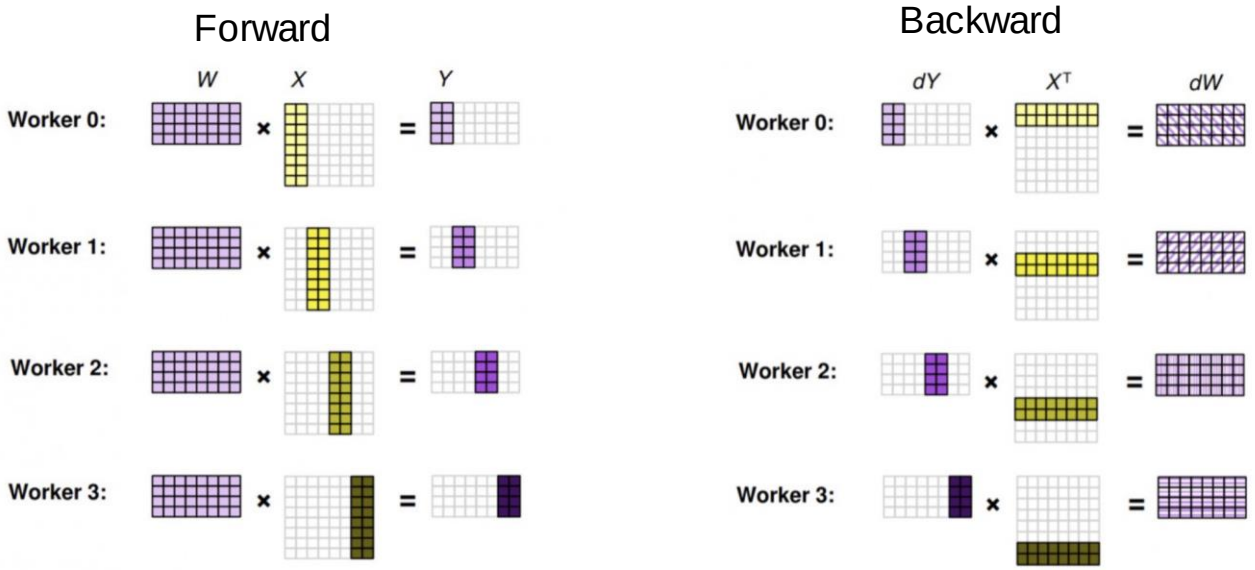
  Распределяем слои/промежуточные вычисления по машинам

- Tensor Parallelism

  Распределяем матричные операции по машинам

Data Parallelism

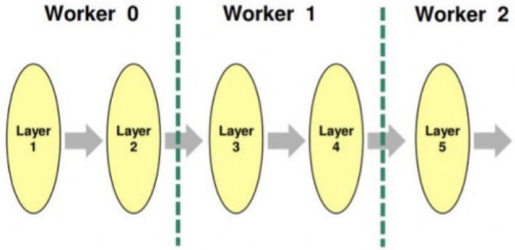
При параллелизме данных входной набор данных разделяется на несколько GPU. Каждый GPU поддерживает полную копию модели и обучается на своем собственном разделе данных, периодически синхронизируя веса с другими GPU



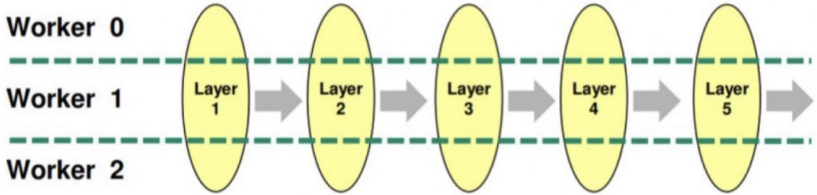
Model Parallelism

Модель разделяется на несколько GPU, при этом каждый графический процессор отвечает только за часть модели.

- Inter-layer Parallel (aka Pipeline Parallel):
  - A worker is responsible for its portion of the layers



- Intra-layer Parallel:
  - A worker is responsible for its portion of each layer



Tensor Parallelism

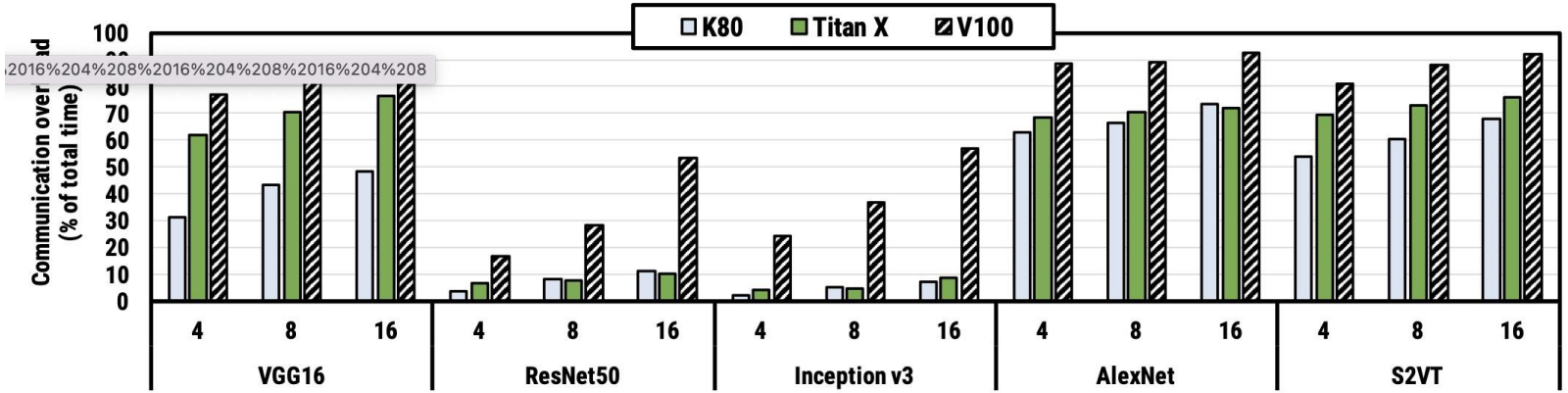
Распределение матричных операций на несколько машин.
Позволяет не хранить всю матрицу полностью на одной машине.
Полезно, когда дорого вычислить один слой на одной машине.
Чаще всего применяется в NLP задачах.

Пример: Shoeybi, Mohammad, et al. "Megatron-lm: Training multi-billion parameter language models using model parallelism." *arXiv preprint arXiv:1909.08053* (2019).

Проблема Data Parallelism: простои GPU из-за пересылки сообщений

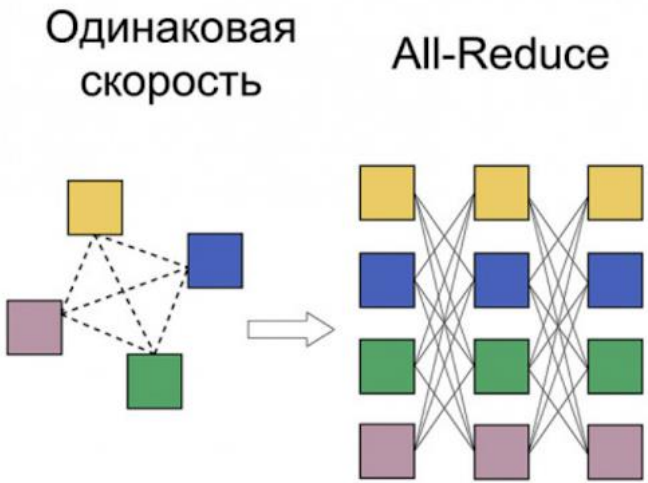
Два фактора способствуют простям:

- Увеличение параллельно работающих GPU
- Увеличение вычислительной мощности GPU



Источник: Harlap, Aaron, et al. "Pipedream: Fast and efficient pipeline parallel dnn training." *arXiv preprint arXiv:1806.03377* (2018).

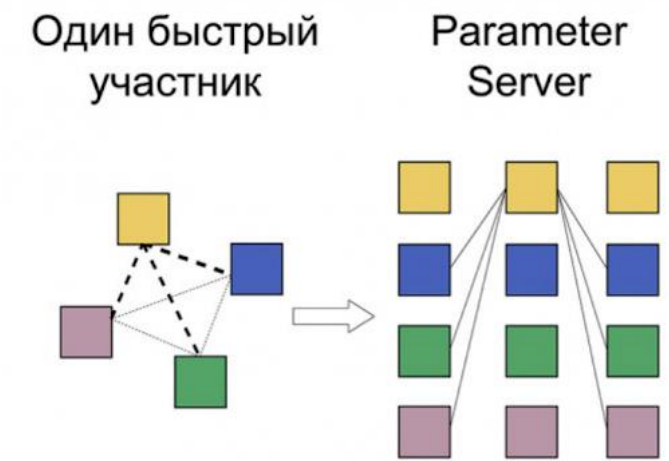
Пути решения: Allreduce



источник: Patarasuk, Pitch, and Xin Yuan. "Bandwidth optimal all-reduce algorithms for clusters of workstations." *Journal of Parallel and Distributed Computing* 69.2 (2009): 117-124.

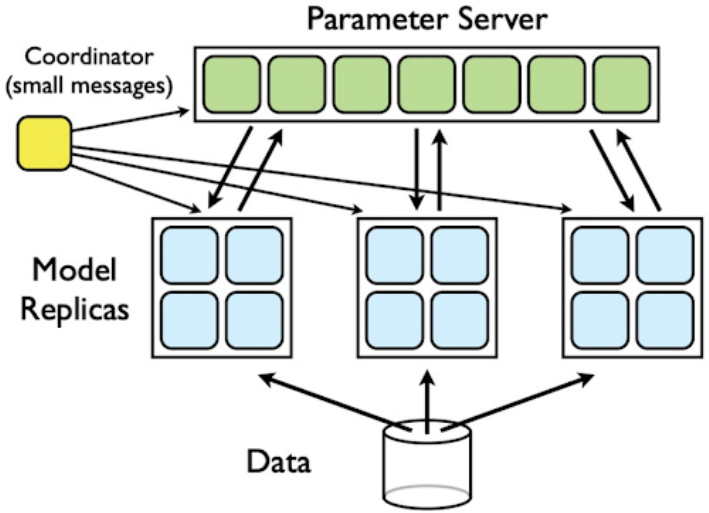
Доклад яндекса (но тут скорее статья для PCa):
https://habr.com/ru/company/yandex/blog/525020/

Пути решения: parameter server



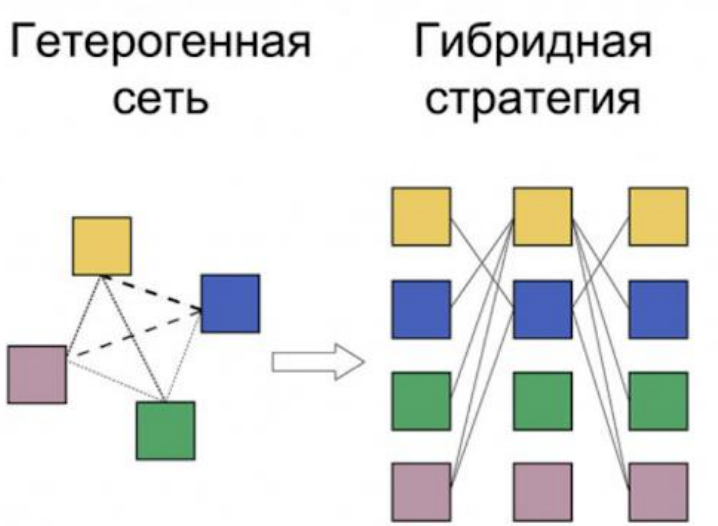
источник: Li, Mu, et al. "Communication efficient distributed machine learning with the parameter server." *Advances in Neural Information Processing Systems* 27 (2014).

Пути решения: coordinator



источник: Dean, Jeffrey, et al. "Large scale distributed deep networks." *Advances in neural information processing systems* 25 (2012).

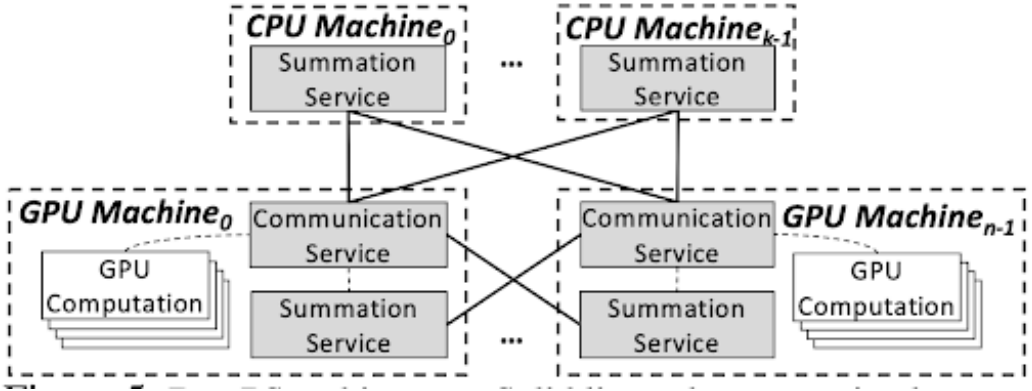
Пути решения: Комбинации
Многопользовательское обучение



источник: Diskin, Michael, et al. "Distributed deep learning in open collaborations." *Advances in Neural Information Processing Systems* 34 (2021): 7879-7897.

описание: https://habr.com/ru/company/yandex/blog/574466/

Пути решения: Комбинации
Решение для вычислительных кластеров



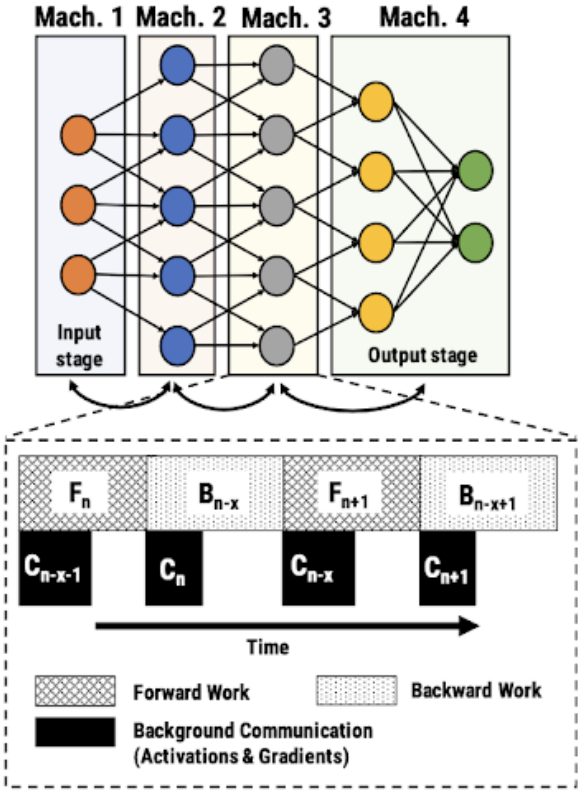
источник: Jiang, Yimin, et al. "A unified architecture for accelerating distributed {DNN} training in heterogeneous {GPU/CPU} clusters." *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 2020.

Современные подходы: Data + Model Parallelism

PipeDream: современная модель Microsoft, объединяющая в себе Data и Model параллелизации

Источник: Harlap, Aaron, et al. "Pipedream: Fast and efficient pipeline parallel dnn training." *arXiv preprint arXiv:1806.03377* (2018).

PipeDream: Pipeline Model Parallelism



На каждую машину загружается несколько слоев нейросети, для каждого из которых выполняется как проход вперед так и проход назад. Каждая машина на каждом шаге выполняет две последовательных операции:

- Forward work
- Backward work

При том пока GPU производит Forward work CPU отправляет и подгружает данные для Backward work, и наоборот.

В любой момент времени на каждом слое производятся forward и backward шаги, а после необходимые данные следующих/предыдущих слоев передаются вперед/назад по цепочке.

PipeDream: Data Parallelism

Каждую машину можно разделять еще на несколько используя Data Parallelism.

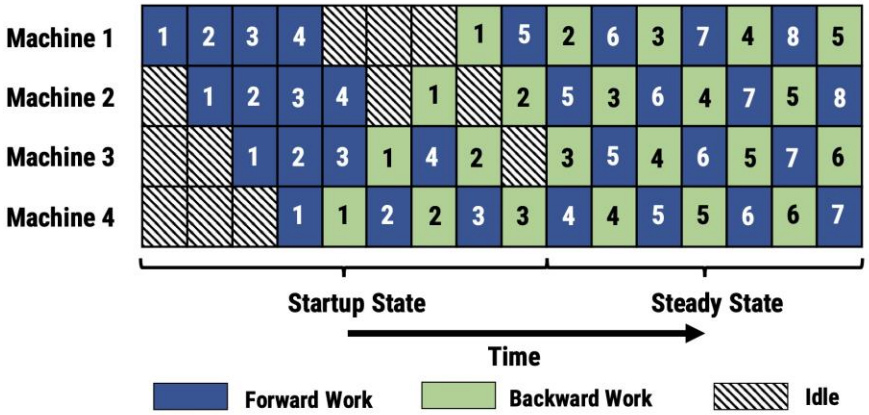
Для синхронизации между машинами на одном этапе используется примитив AllReduce.

PipeDream: распределение слоев по машинам

При кажущейся тривиальности распределение может очень сильно влиять на общую производительность, ведь скорость продвижения данных по конвейеру равна скорости обработки очередного шага на самой медленной из машин на каждом этапе.

Задача решается методом **динамического программирования.**

PipeDream: проблема неактуальности градиента



$$w^{(t+1)} = w^{(t)} - \nu \cdot \nabla f(w_1^{(t-n+1)}, w_2^{(t-n+2)}, \ldots, w_n^{(t)})$$

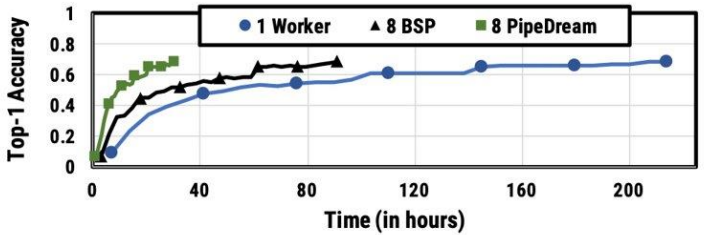
где w = вектор всех параметров, а w_i вектор параметров вычисляемых на машине номер i.

PipeDream: тесты

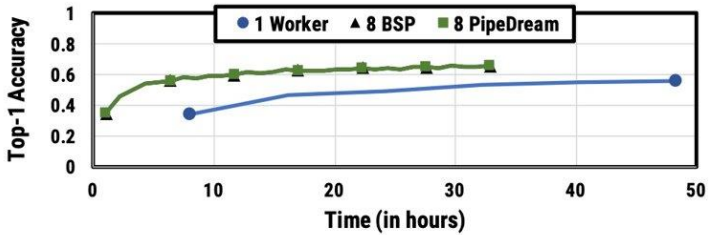
DNN Model	# Machines (Cluster)	BSP speedup over 1 machine	PipeDream Config	PipeDream speedup over 1 machine	PipeDream speedup over BSP	PipeDream communication reduction over BSP
VGG16	4 (A)	1.47×	2-1-1	3.14×	2.13×	90%
	8 (A)	2.35×	7-1	7.04×	2.99×	95%
	16 (A)	3.28×	9-5-1-1	9.86×	3.00×	91%
	8 (B)	1.36×	7-1	6.98×	5.12×	95%
Inception-v3	8 (A)	7.66×	8	7.66×	1.00×	0%
	8 (B)	4.74×	7-1	6.88×	1.45×	47%
S2VT	4 (A)	1.10×	2-1-1	3.34×	3.01×	95%

Сводка результатов сравнения PipeDream с конфигурациями с параллельными данными (BSP) при обучении моделей до заявленной окончательной точности. «Конфигурация PipeDream» представляет собой конфигурацию, сгенерированную нашим алгоритмом разделения, например, «2-1-1» — это конфигурация, в которой модель разделена на три этапа, причем первый этап реплицируется на двух машинах.

PipeDream: тесты



(a) VGG16



(b) Inception-v3

Accuracy vs. time for VGG16 and Inception-v3 with 8 machines on Cluster-A

Резюме

- Если хочется сэкономить память, необязательно использовать параллелизм модели, попробуй сначала checkpointing и вычисления с меньшей битностью
- Распараллелить моделю можно тремя способами: Data Parallel, Model parallel и Tensor Parallel
- Самый популярный и сам простой метод: Data Parallelism
- Итоговая архитектура параллельного обучения нейронной сети сильно зависит от решаемой задачи, нету единого решения
- Для достижения максимальной производительности как правило приходится использовать комбинации различных подходов.


Faculty
of
Computer
science
Higher School of Economics