Model-Parallel Deep Learning

Episode III, YSDA'21

Yandex
Research

LAMBDA

British Hedgehog
Preservation Society

Training large DL models

~~Model-Parallel Deep Learning~~

Episode III, YSDA'21

Yandex
Research

LAMBDA

British Hedgehog
Preservation Society

Recap: large models

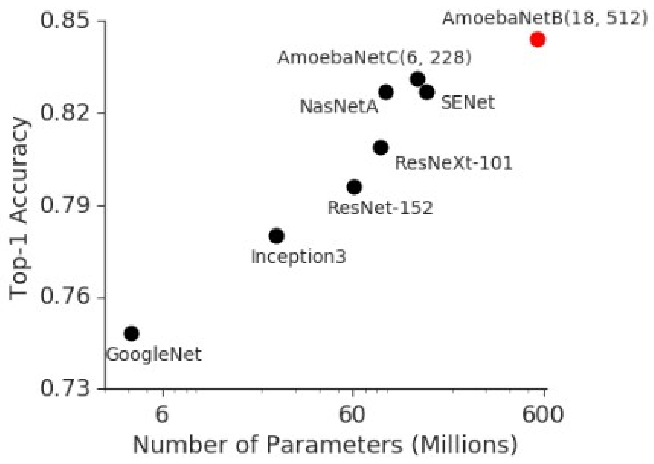
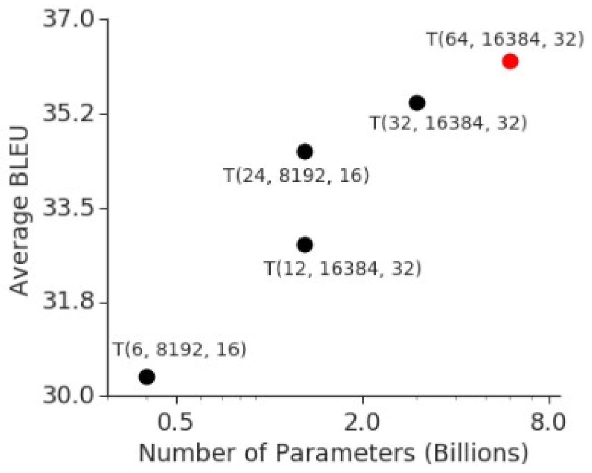


Image Classification
ImageNet

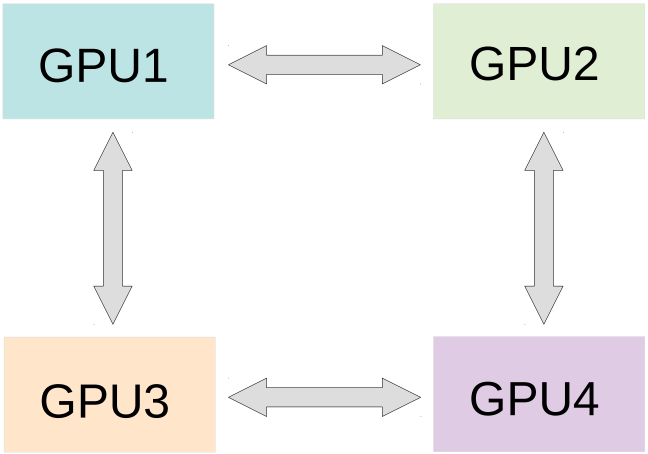


Machine Translation
average over WMT

Source: https://arxiv.org/abs/1811.06965

Recap: Ring allreduce

Bonus quest: you can only send data between **adjacent** gpus



Ring topology

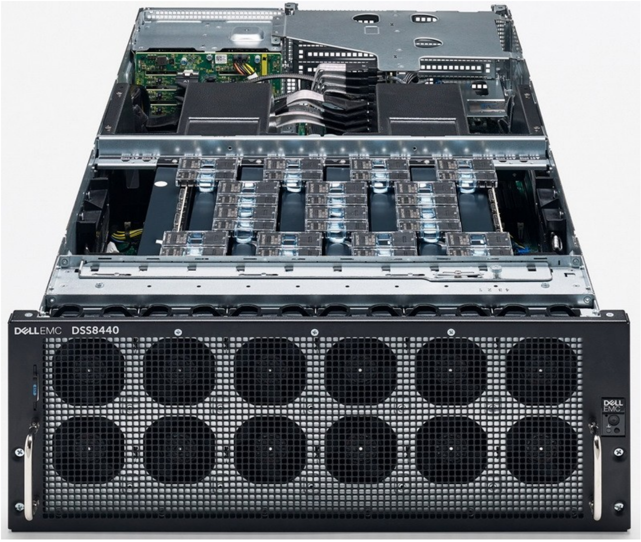


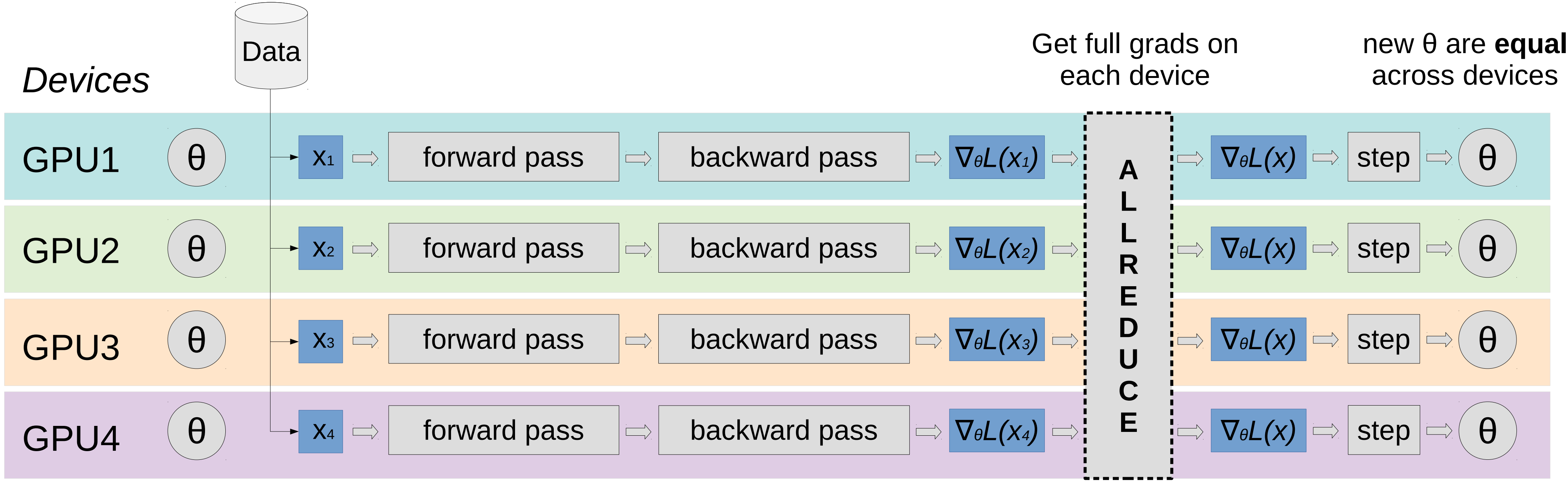
Image: graphcore ipu server

Answer & more: tinyurl.com/ring-allreduce-blog

Recap: All-Reduce SGD

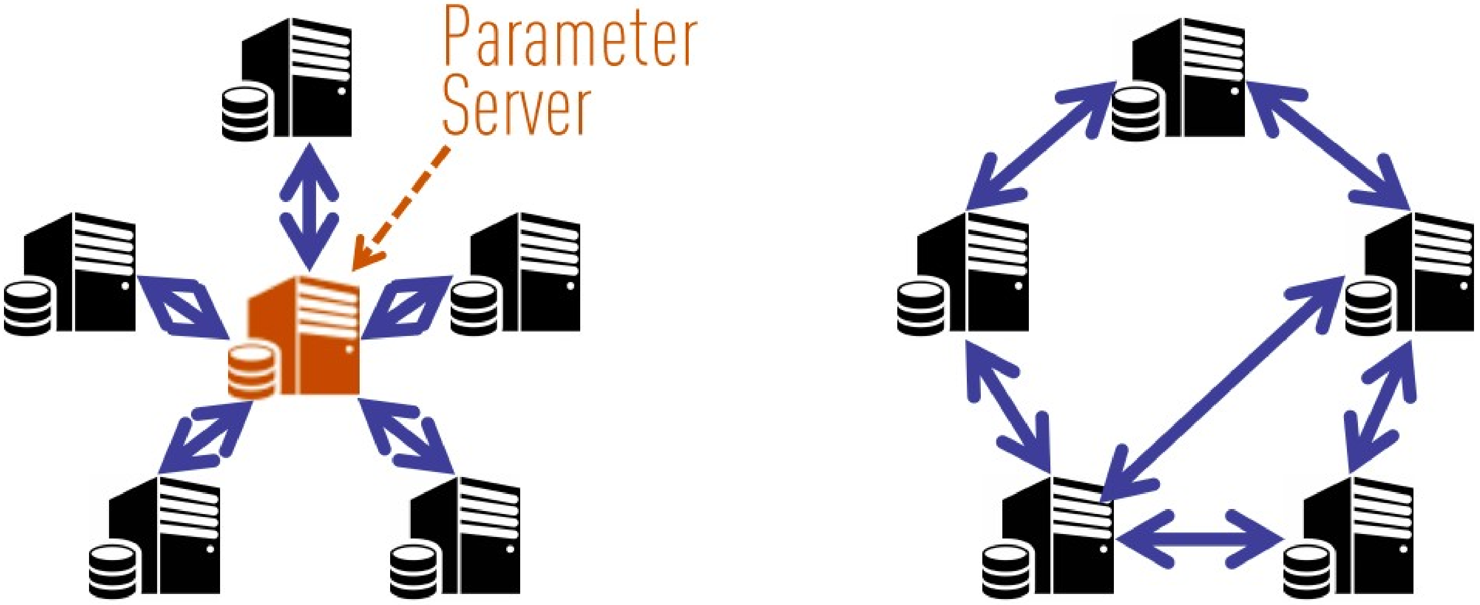
arxiv.org/abs/1706.02677

Idea: get rid of the host, each gpu runs its own computation
Q: why will weights be equal after such step?



Recap: Decentralized SGD

Gossip (communication): https://tinyurl.com/boyd-gossip-2006
Gossip outperforms All-Reduce: https://tinyurl.com/can-dsgd-outperform



(a) Centralized Topology (b) Decentralized Topology

Q: What if a model is larger than GPU?

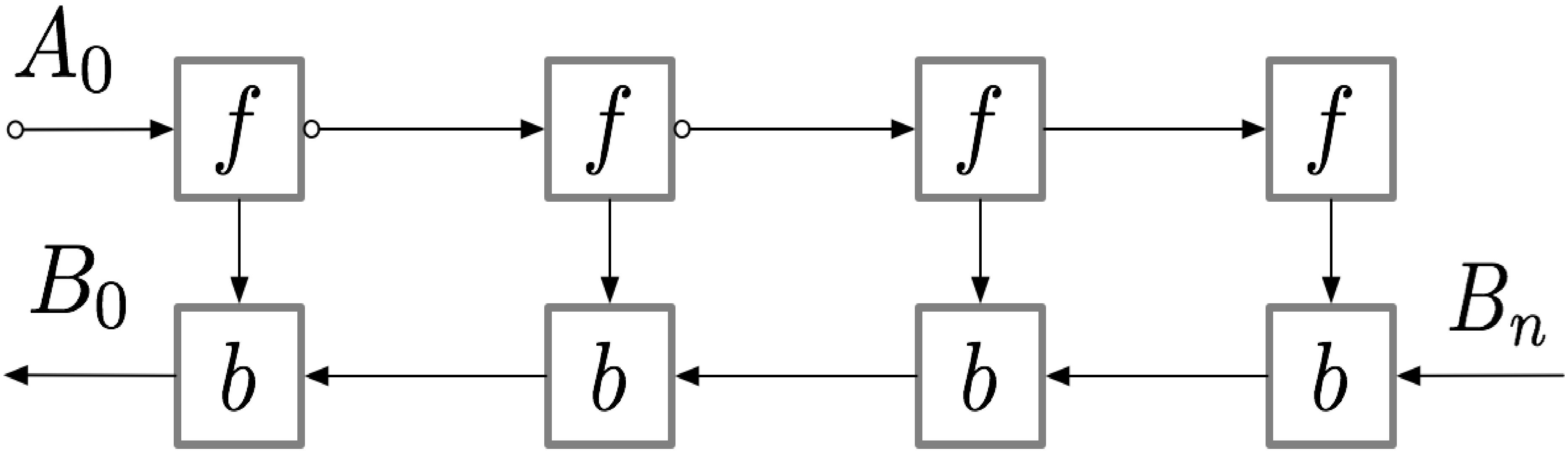
Q: What if a model is larger than GPU?

easy mode: cannot fit batch size 1

expert mode: not even parameters!

Gradient checkpointing

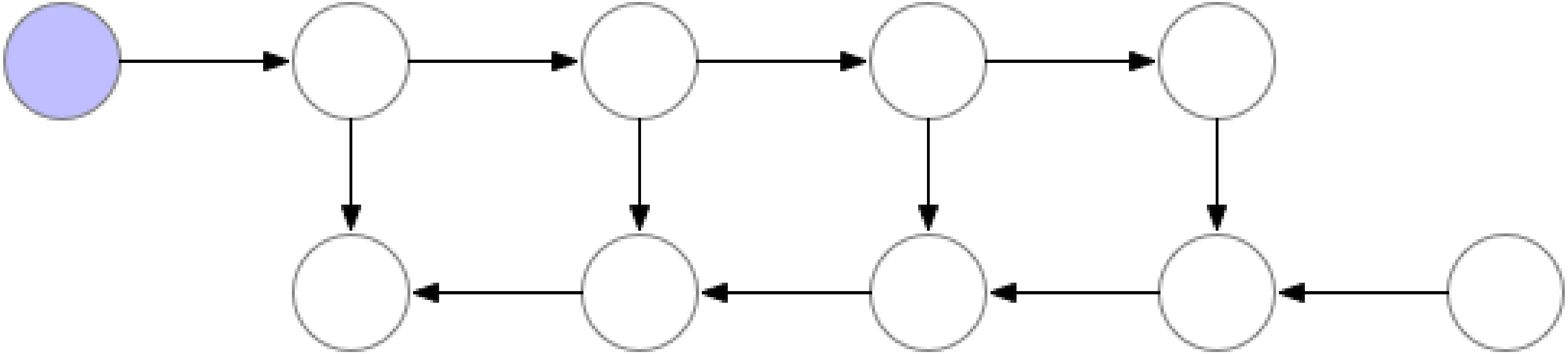
aka rematerialization



Paper (DL): arxiv.org/pdf/1604.06174.pdf
TF: github.com/cybertronai/gradient-checkpointing
Pytorch: pytorch.org/docs/stable/checkpoint.html

Gradient checkpointing

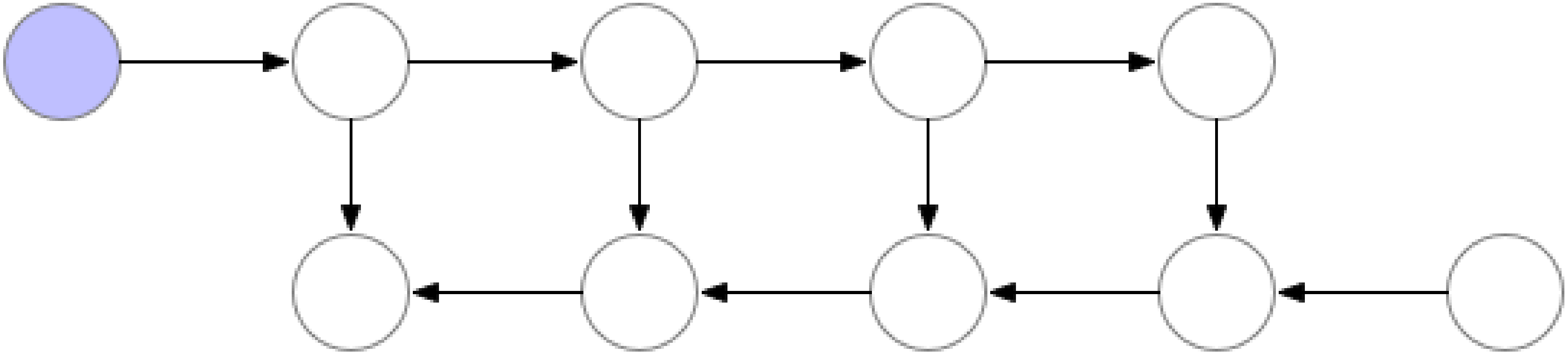
Normal backprop



Paper (DL): arxiv.org/pdf/1604.06174.pdf
TF: github.com/cybertronai/gradient-checkpointing
Pytorch: pytorch.org/docs/stable/checkpoint.html

Gradient checkpointing

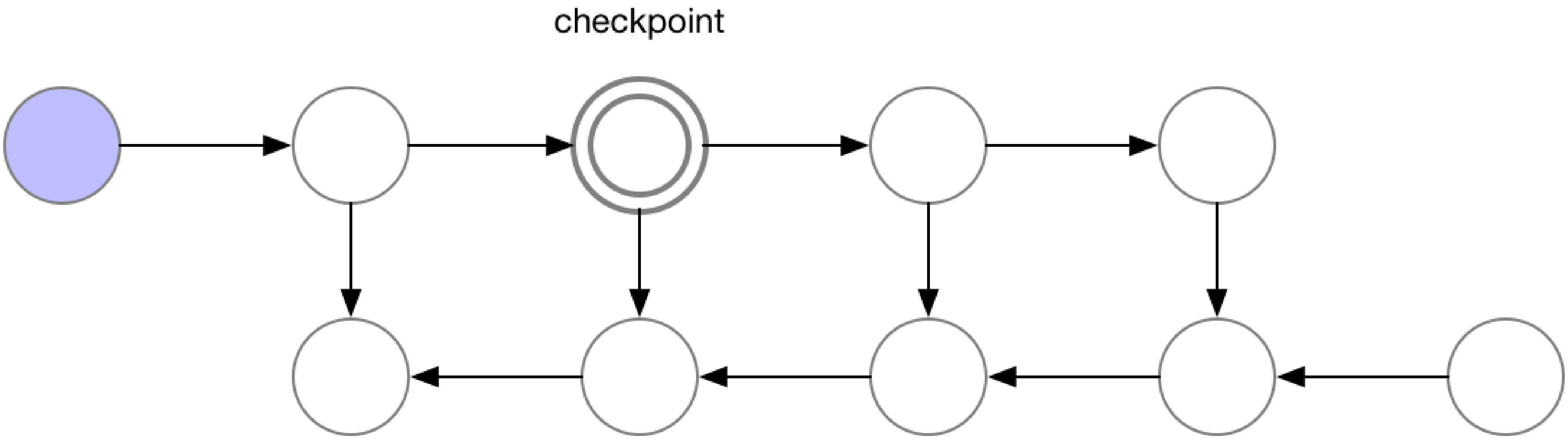
Full rematerialization



Paper (DL): arxiv.org/pdf/1604.06174.pdf
TF: github.com/cybertronai/gradient-checkpointing
Pytorch: pytorch.org/docs/stable/checkpoint.html

Gradient checkpointing

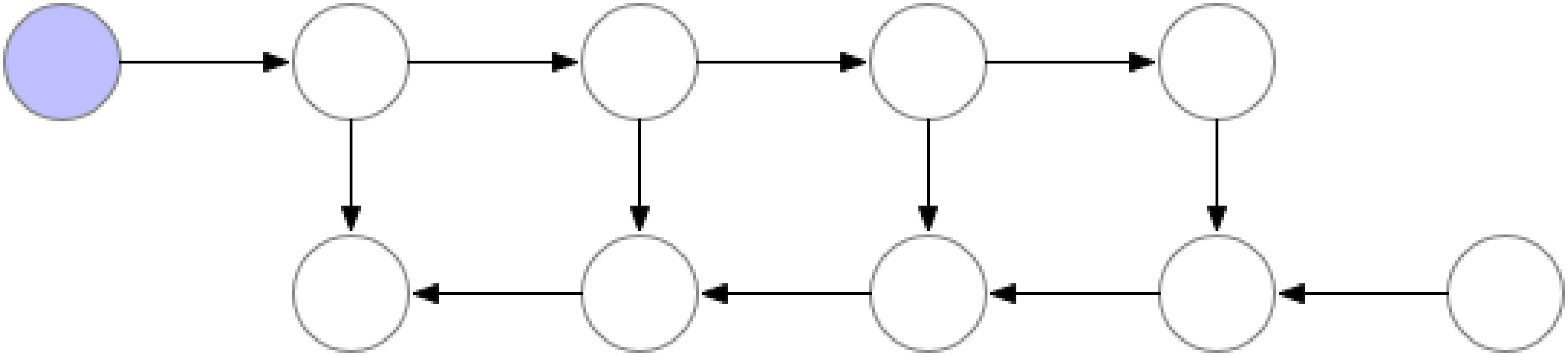
Single checkpoint



Paper (DL): arxiv.org/pdf/1604.06174.pdf
TF: github.com/cybertronai/gradient-checkpointing
Pytorch: pytorch.org/docs/stable/checkpoint.html

Gradient checkpointing

Single checkpoint



Paper (DL): arxiv.org/pdf/1604.06174.pdf
TF: github.com/cybertronai/gradient-checkpointing
Pytorch: pytorch.org/docs/stable/checkpoint.html

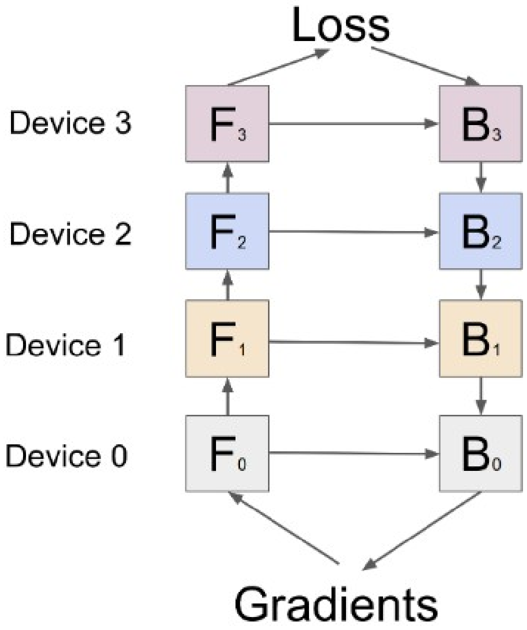
Q: What if a model is larger than GPU?

easy mode: cannot fit batch size 1

expert mode: not even parameters!

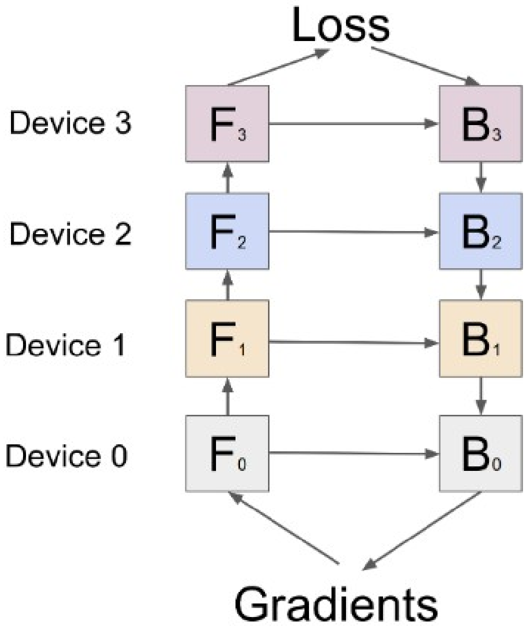
Model-parallel training

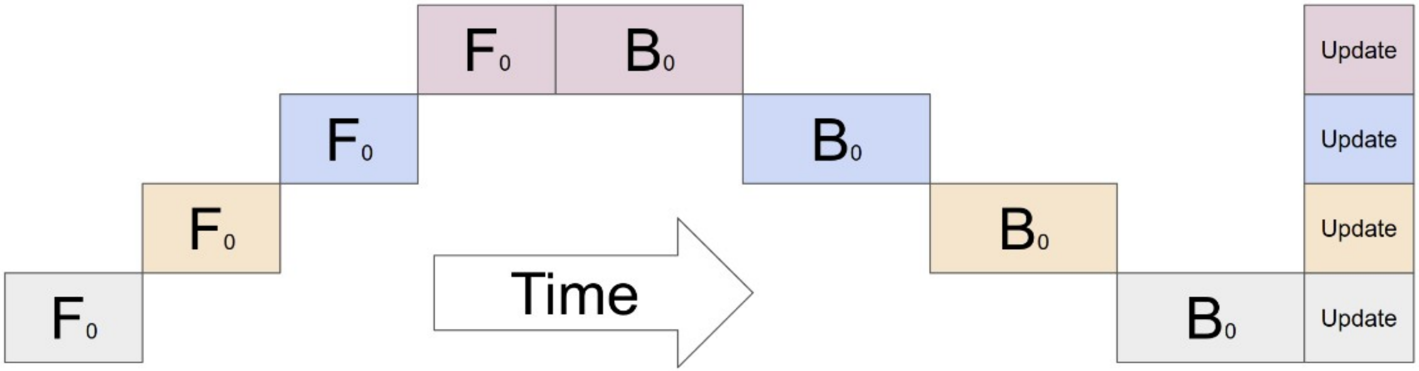
Q: What if a model is larger than GPU?



Model-parallel training

Q: What if a model is larger than GPU?





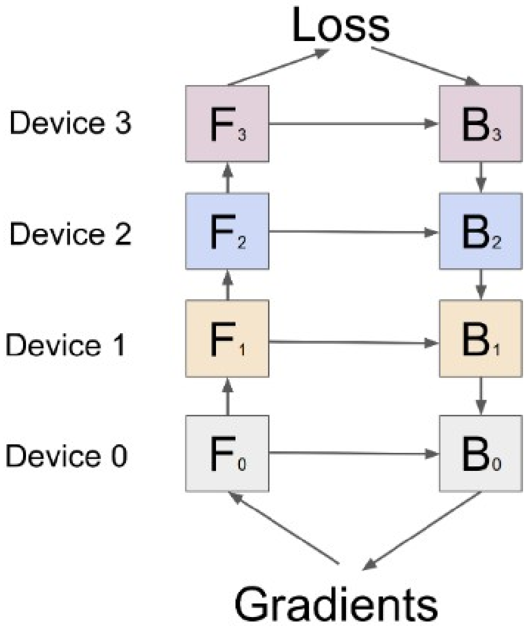
model size: O(N)
throughput: O(1)

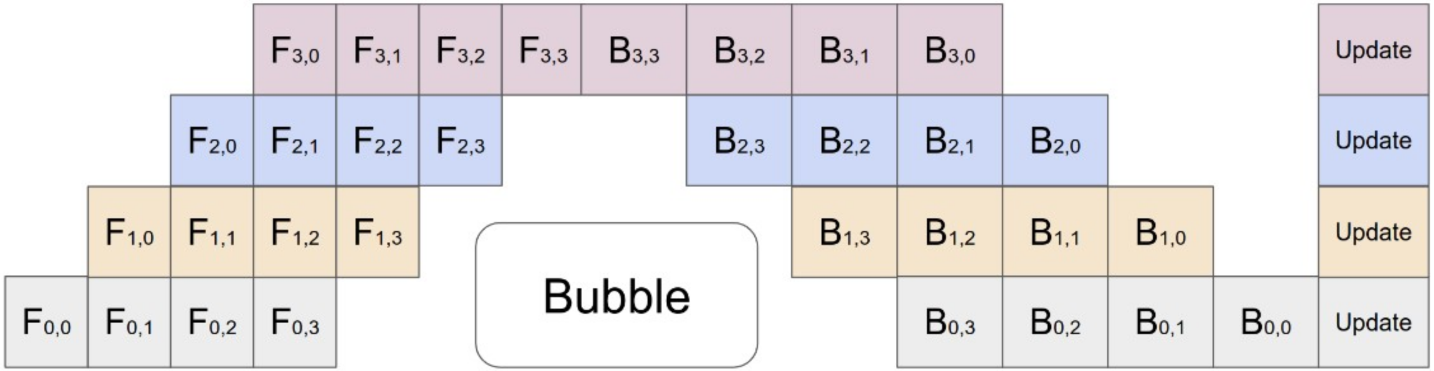
Q: Can we go faster?

Pipelining

GPipe: arxiv.org/abs/1811.06965 – good starting point, *not* the 1st paper

Idea: split data into micro-batches and form a pipeline (right)



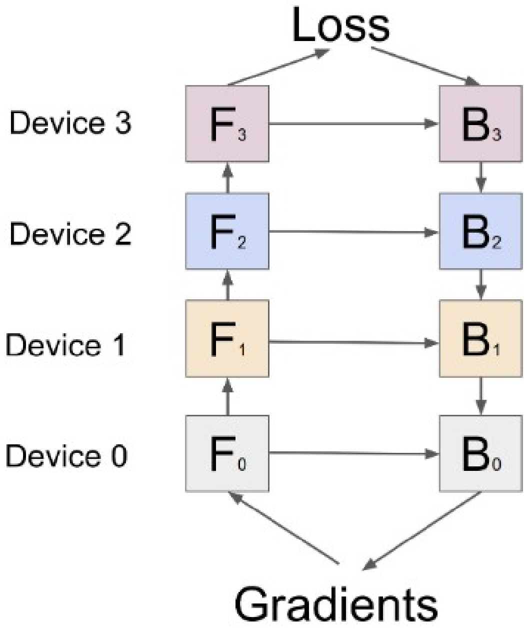


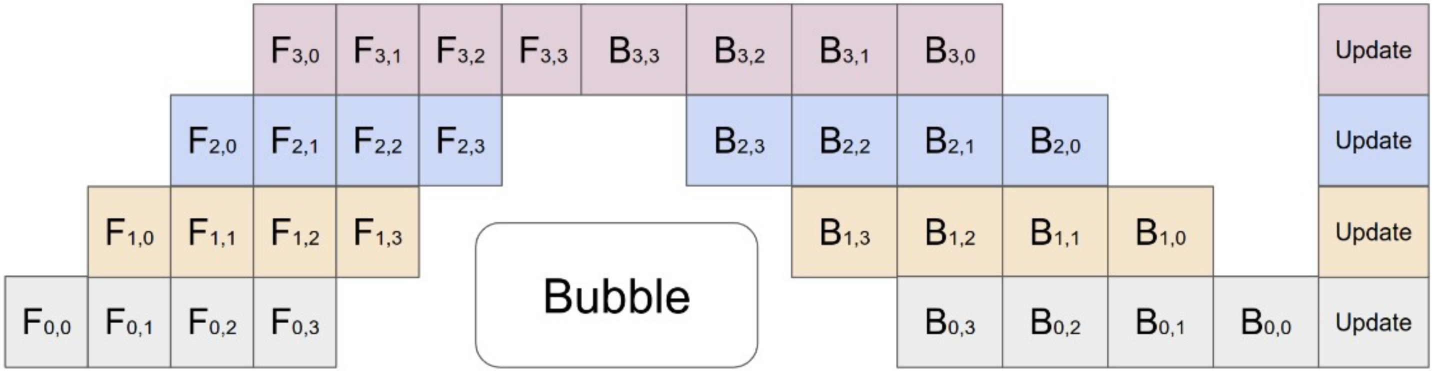
model size: O(n)
throughput: O(n) – with caveats

Pipelining

GPipe: arxiv.org/abs/1811.06965 – good starting point, *not* the 1st paper

Idea: split data into micro-batches and form a pipeline (right)





model size: O(n)
throughput: O(n) – with caveats

Q: Even faster?

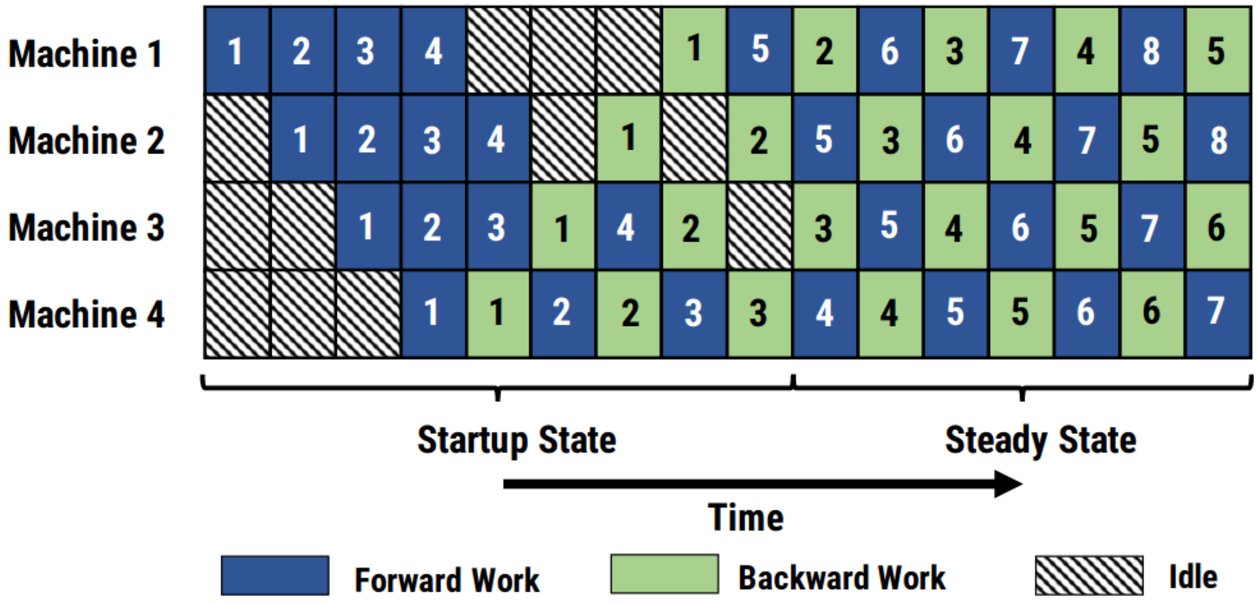
Pipeline-parallel training

PipeDream: arxiv.org/abs/1806.03377

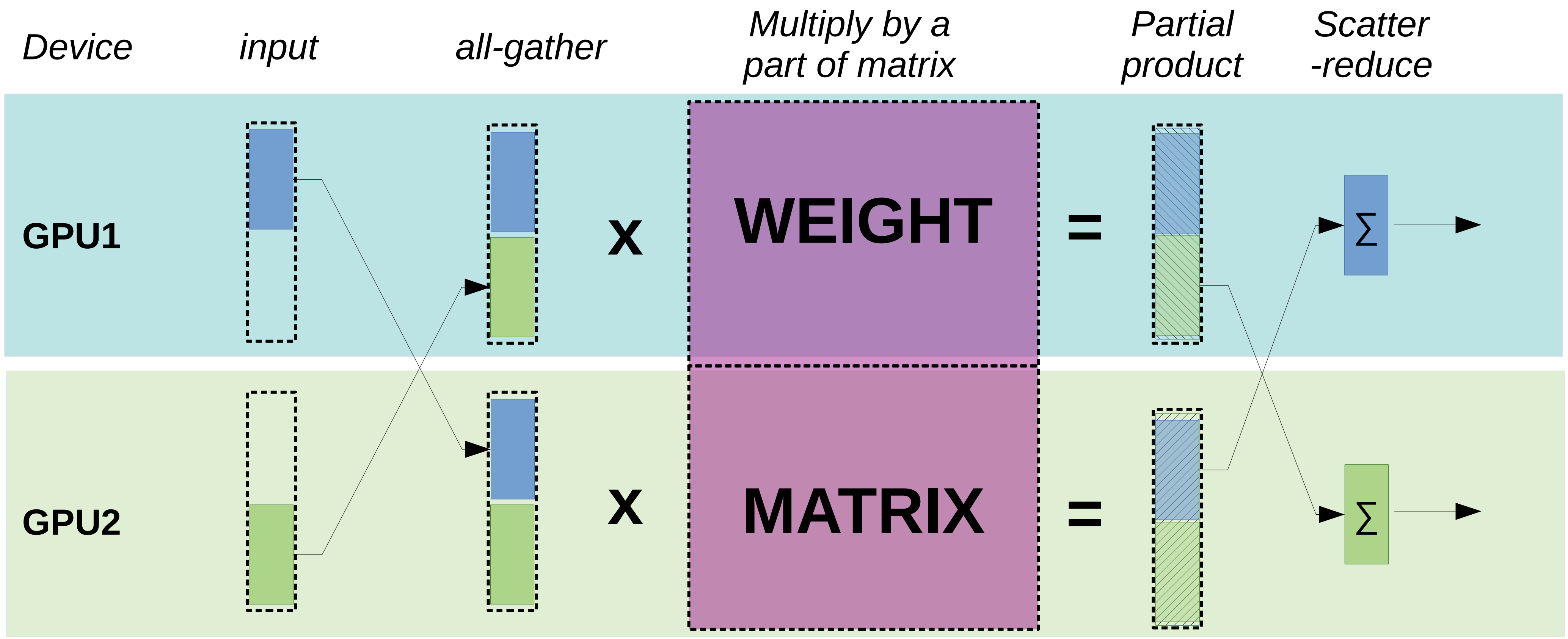
Idea: apply gradients with every microbatch for maximum throughput

Also neat:

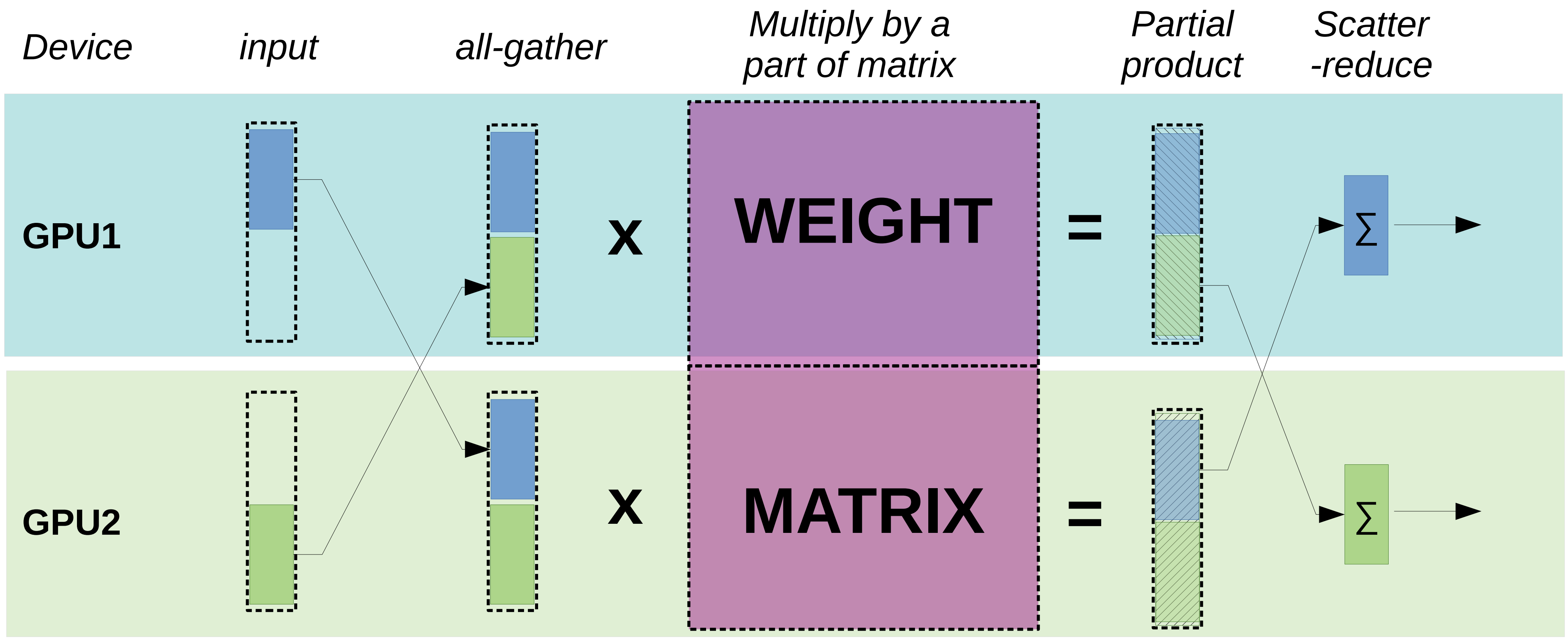
- Automatically partition layers to GPUs via dynamic programming
- *Store k* past weight versions to reduce gradient staleness
- Aims at high latency



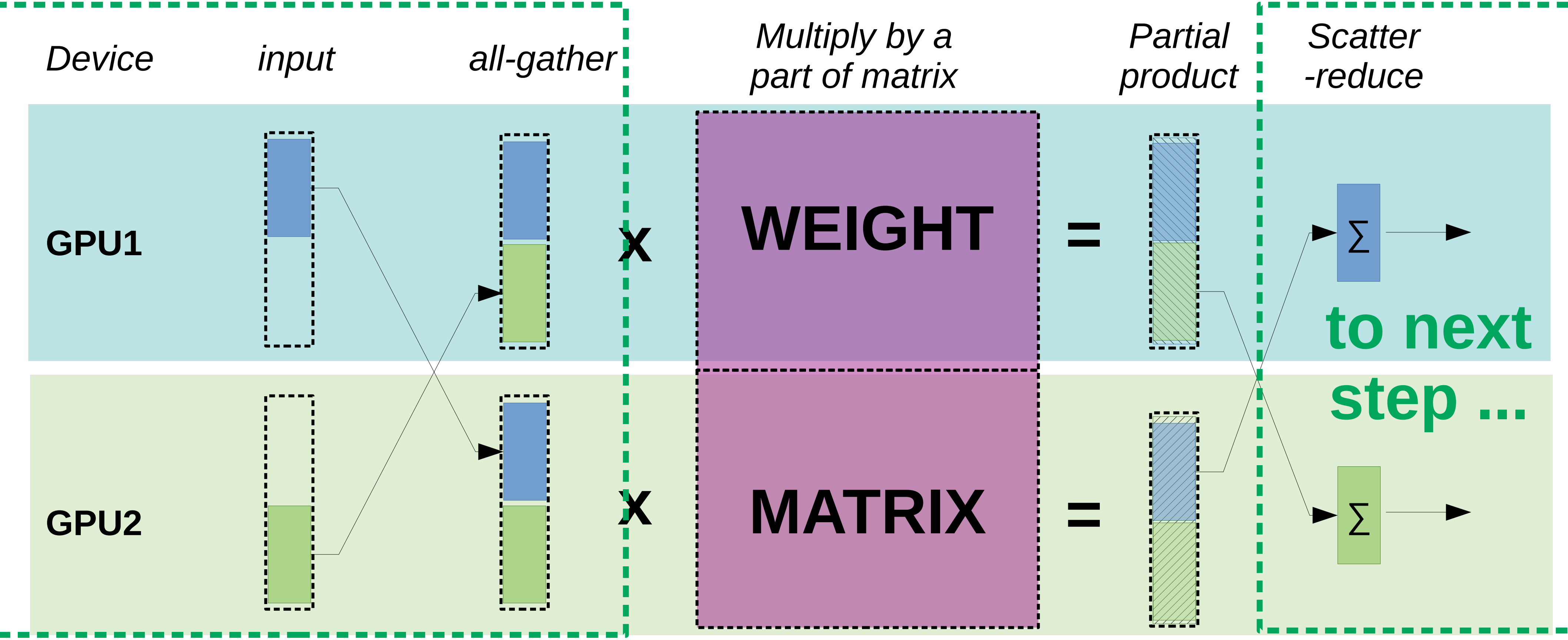
Tensor-parallel training



Q: find AllReduce op here



Q: find AllReduce op here



</Model-parallel>

+ model larger than GPU
+ faster for small
***** typical size: 2-8 gpus
- model partitioning is tricky
tensor parallelism is easier, but requires ultra low latency
- latency is critical, go buy nvlink
except for PipeDream
- *often combined with gradient checkpointing*

Tutorials:

- Simple pipelining in PyTorch – tinyurl.com/pytorch-pipelining
- Distributed model-parallel with torch RPC - https://tinyurl.com/torch-rpc
- Tensor parallelism – *mesh tensorflow* - *arxiv.org/abs/1811.02084*

(more libs in the next section)

</Model-parallel>

+ model larger than GPU
+ faster for small
***** typical size: 2-8 gpus
- model partitioning is tricky
tensor parallelism is easier, but requires ultra low latency
- latency is critical, go buy nvlink
except for PipeDream
- *often combined with gradient checkpointing*

Tutorials:

- Simple pipelining in PyTorch – tinyurl.com/pytorch-pipelining
- Distributed model-parallel with torch RPC - https://tinyurl.com/torch-rpc
- Tensor parallelism – *mesh tensorflow* - *arxiv.org/abs/1811.02084*

Q: what if you have 1024 GPUs, but the model fits on 8?

</Model-parallel>

+ model larger than GPU
+ faster for small
***** typical size: 2-8 gpus
- model partitioning is tricky
tensor parallelism is easier, but requires ultra low latency
- latency is critical, go buy nvlink
except for PipeDream
- *often combined with gradient checkpointing*

Tutorials:

- Simple pipelining in PyTorch – tinyurl.com/pytorch-pipelining
- Distributed model-parallel with torch RPC - https://tinyurl.com/torch-rpc
- Tensor parallelism – *mesh tensorflow* - *arxiv.org/abs/1811.02084*

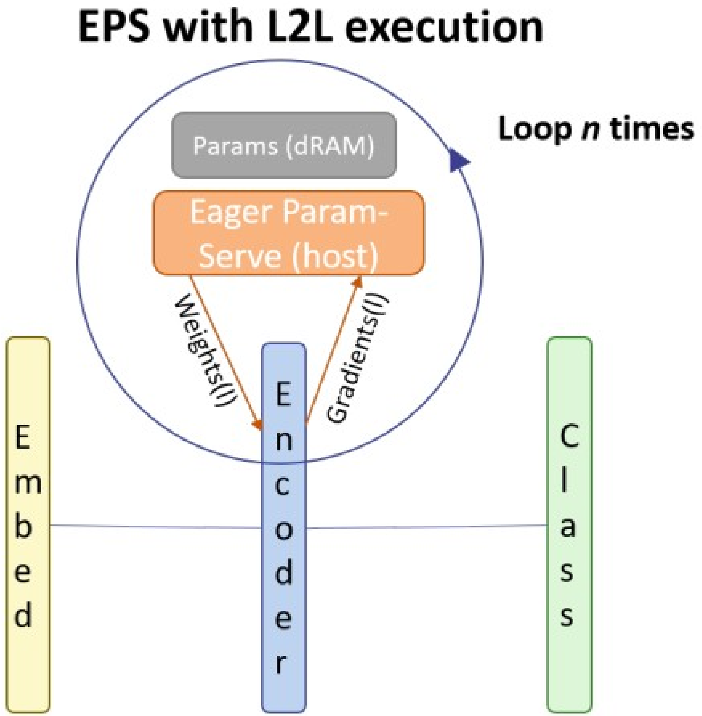
Large-scale training: combine model- and data-parallel

Case study: DeepSpeed

Source: microsoft

Memory offloading

L2L: https://arxiv.org/abs/2002.05645



- Initialize all layers on CPU
- Move k layers at a time to GPU
- Remove layers after computation

- Fetch k+1-st layer while k-th runs
- Still 20-50% overhead

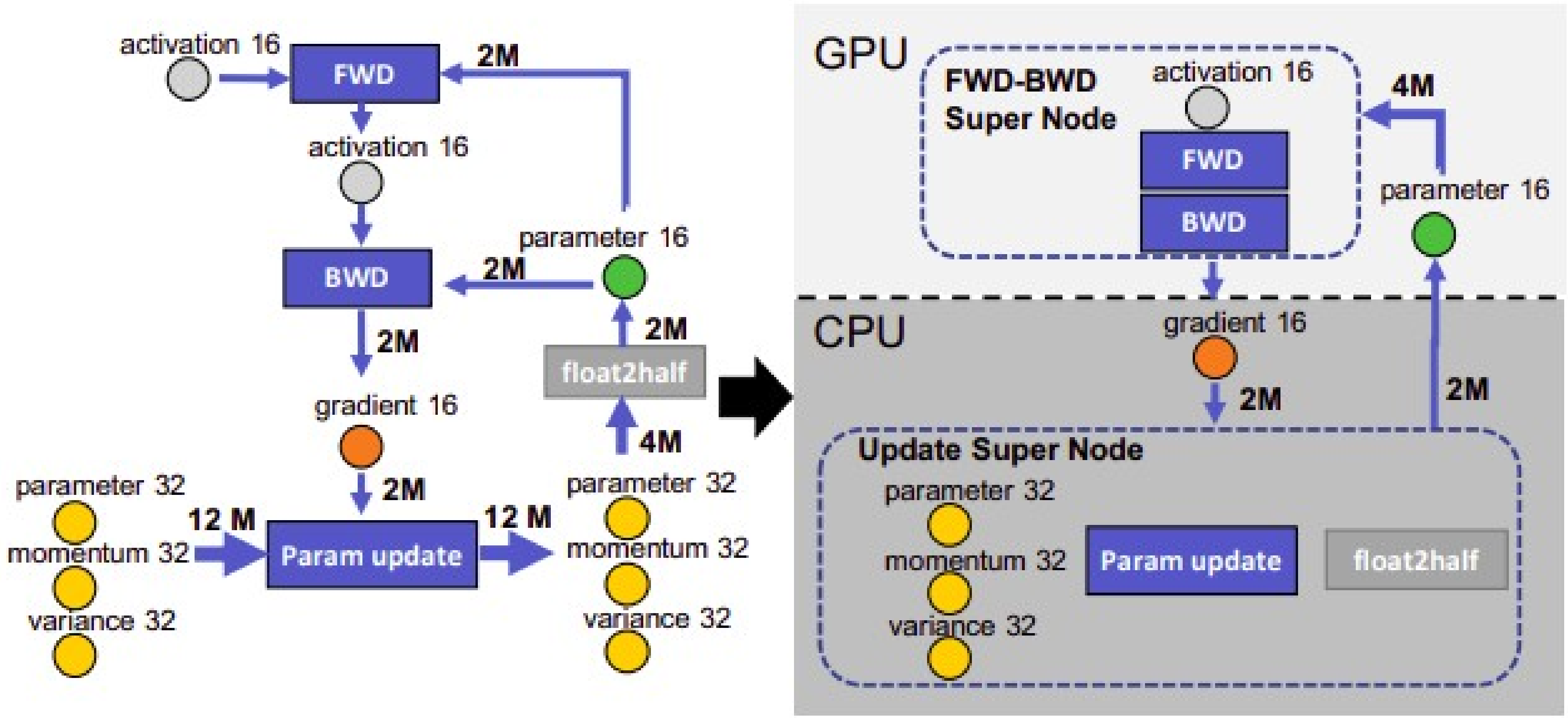
Memory offloading

L2L: https://arxiv.org/abs/2002.05645

Method	uBatch Size	Device Batch Size	#Layer	#parameters	Memory (GB)
Baseline	2	2	24	300 Million	9.23
Baseline	2	**2**	**48**	600 Million	**OOM**
L2L-stash on GPU	64	64	24	300 Million	5.22
L2L-stash on GPU	**64**	**64**	**48**	600 Million	**6.76**
L2L-stash on GPU	**64**	**64**	**96**	1.2 Billion	**9.83**
L2L-stash on CPU	64	64	24	300 Million	3.69
L2L-stash on CPU	**64**	**64**	**96**	1.2 Billion	**3.69**
L2L-stash on CPU	**64**	**64**	**384**	4.8 Billion	**3.69**

Memory offloading

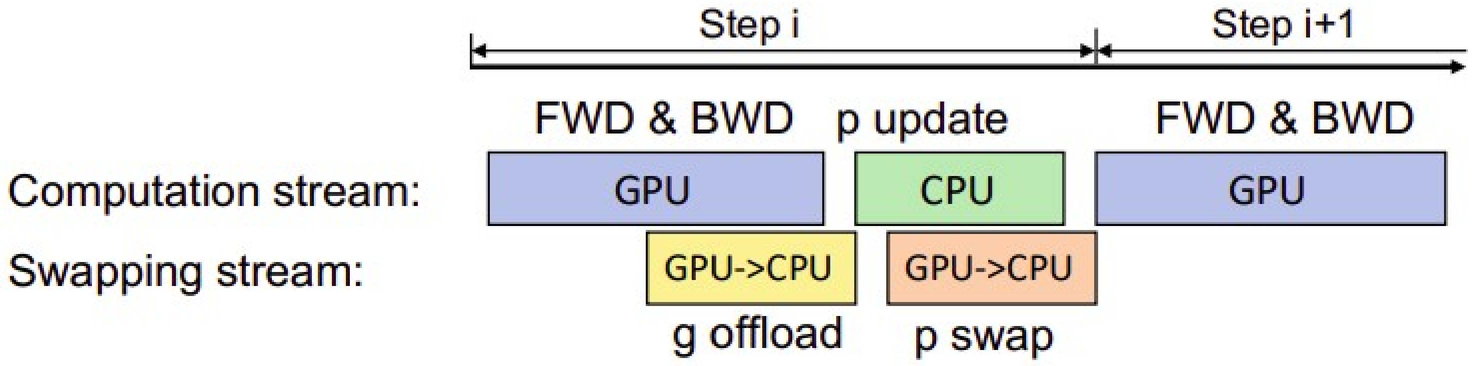
ZeRO-offload: https://arxiv.org/abs/2101.06840



Memory offloading

ZeRO-offload: https://arxiv.org/abs/2101.06840

- Offload **in parallel** with computation
- Use gradient checkpointing
- Delayed parameter update



Memory offloading

ZeRO-offload: https://arxiv.org/abs/2101.06840

- Offload in parallel with computation
- Use gradient checkpointing
- **Delayed parameter update**

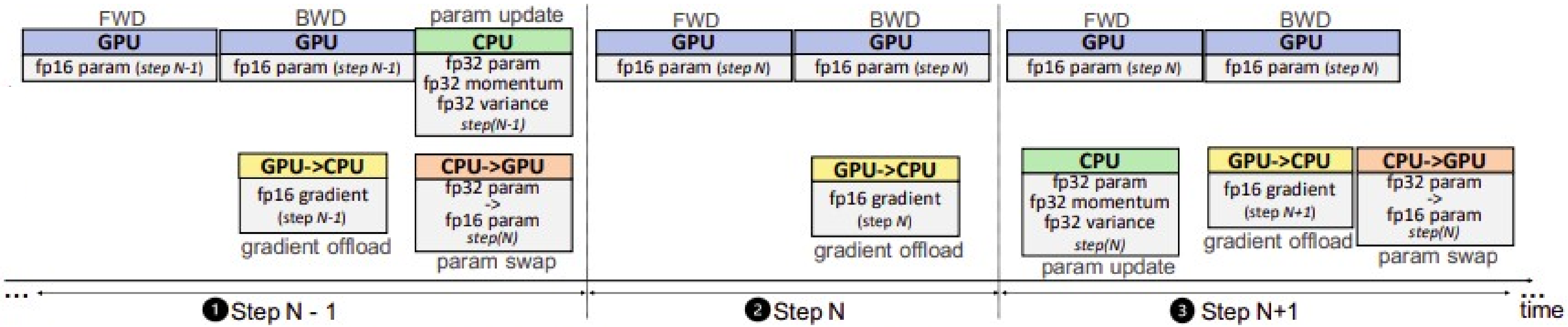
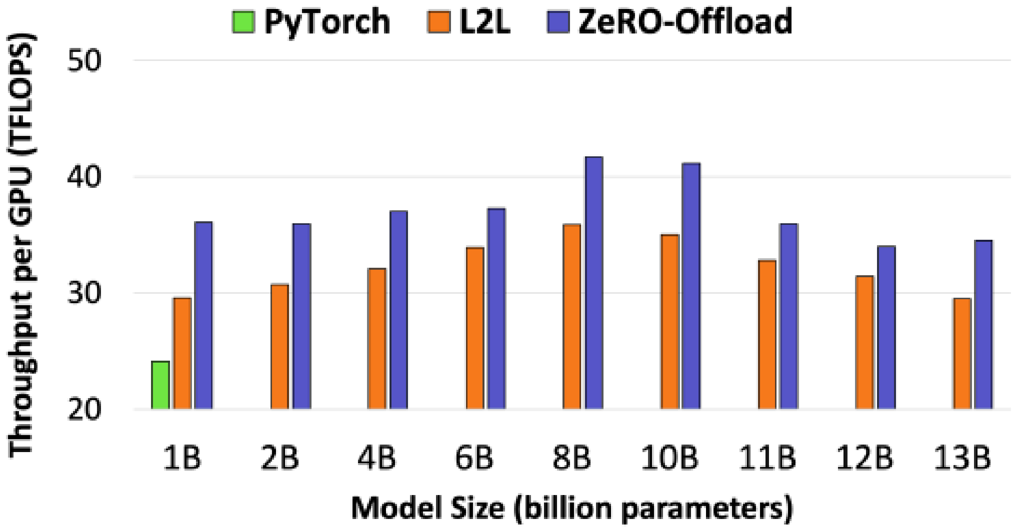


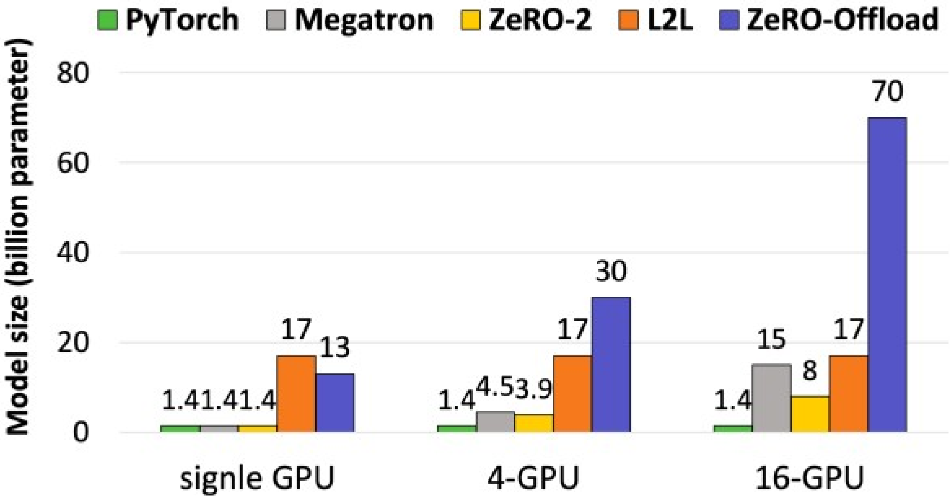
Figure 6: Delayed parameter update during the training process.

Memory offloading

ZeRO-offload: https://arxiv.org/abs/2101.06840

- Offload in parallel with computation
- Use gradient checkpointing
- Delayed parameter update

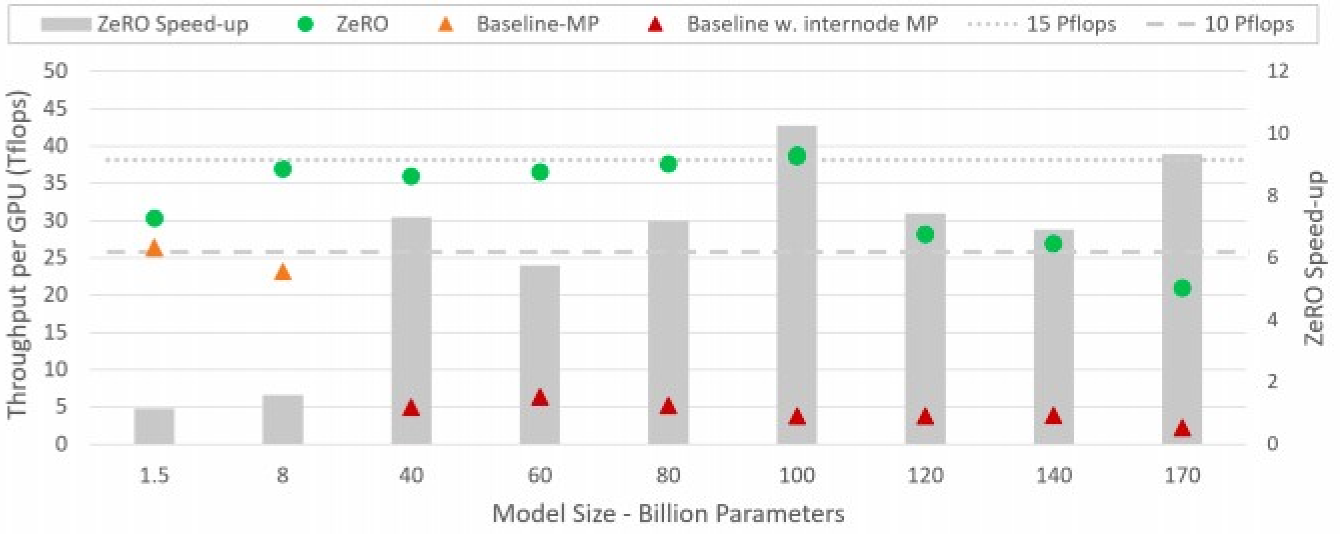




DeepSpeed / ZeRO

ZeRO: https://arxiv.org/pdf/1910.02054v3.pdf

- Combines sharded DPP and offload
- … and some tensor parallelism
- … and a ton of hacks



</ZeRO>

Multi-GPU strategies:
* Pipeline model-parallel – allocate layers on different GPUs
* Sharded data-parallel – split optimizer state and/or parameters

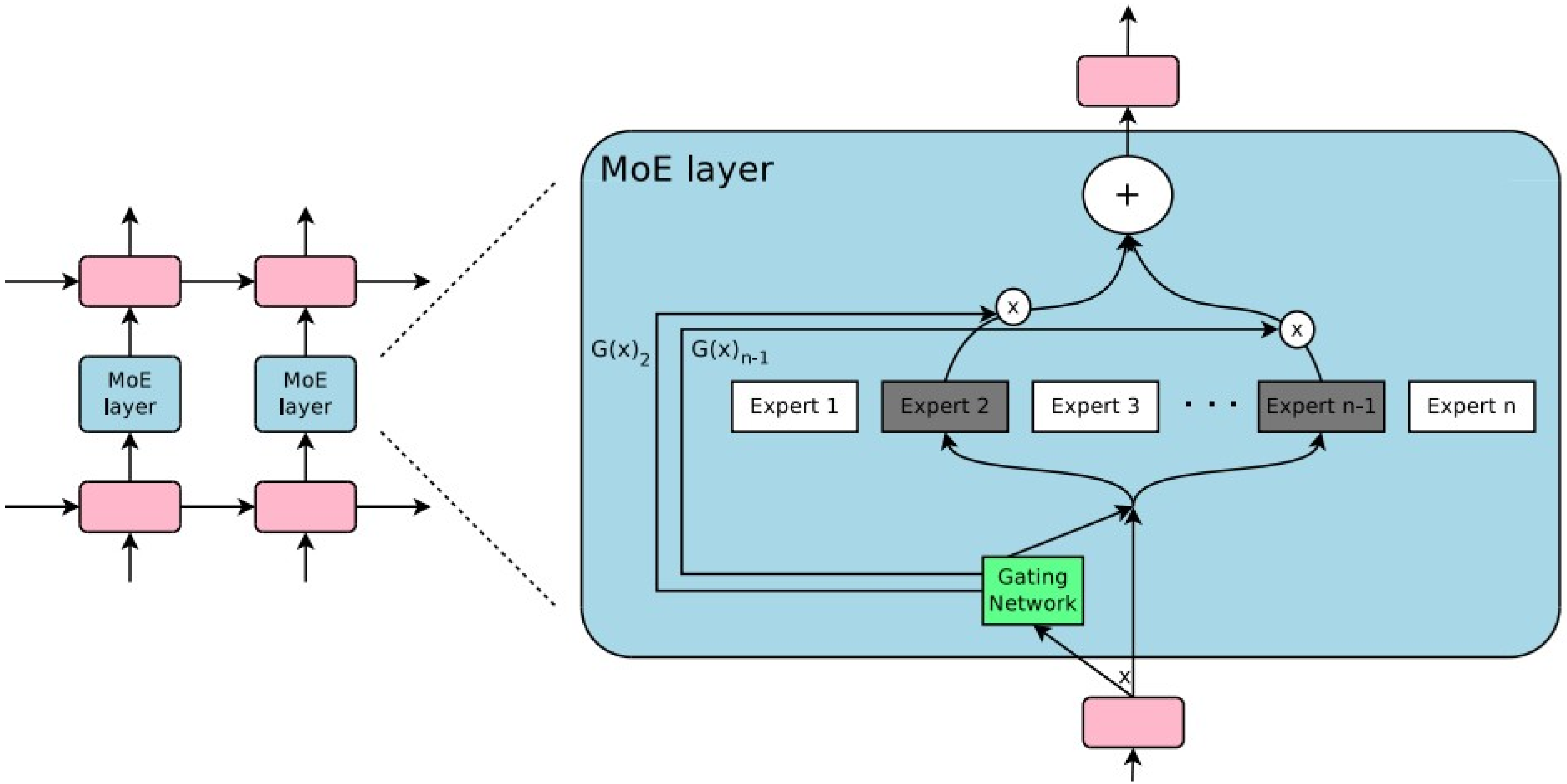
Single GPU strategies:
* Small model – gradient checkpointing & virtual batch
* Large model – optimizer state sharding (keep parameters on GPU)

Implementations:

- DeepSpeed– sharded DP, offload, tensor parallelism, active development
  - Offload – https://www.deepspeed.ai/news/2021/03/07/zero3-offload.html
- Fairscale – most of DeepSpeed features with friendrier API
  - One great implementation – https://github.com/NVIDIA/Megatron-LM

Expert Parallelism

Sparsely gated MoE: https://arxiv.org/pdf/1701.06538.pdf

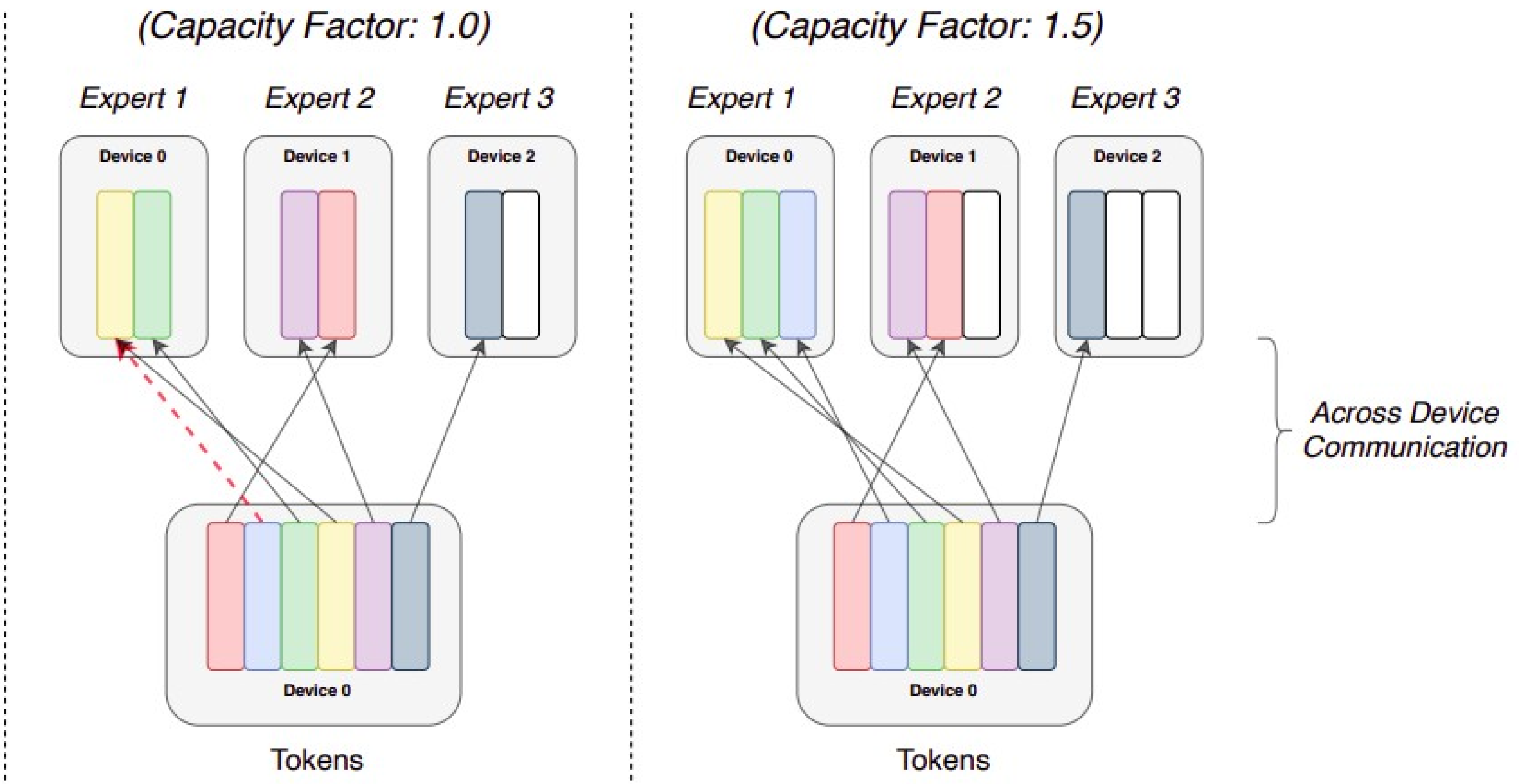


MoE Variant: Switch Transformer

Switch: https://arxiv.org/pdf/2101.03961.pdf

Terminology

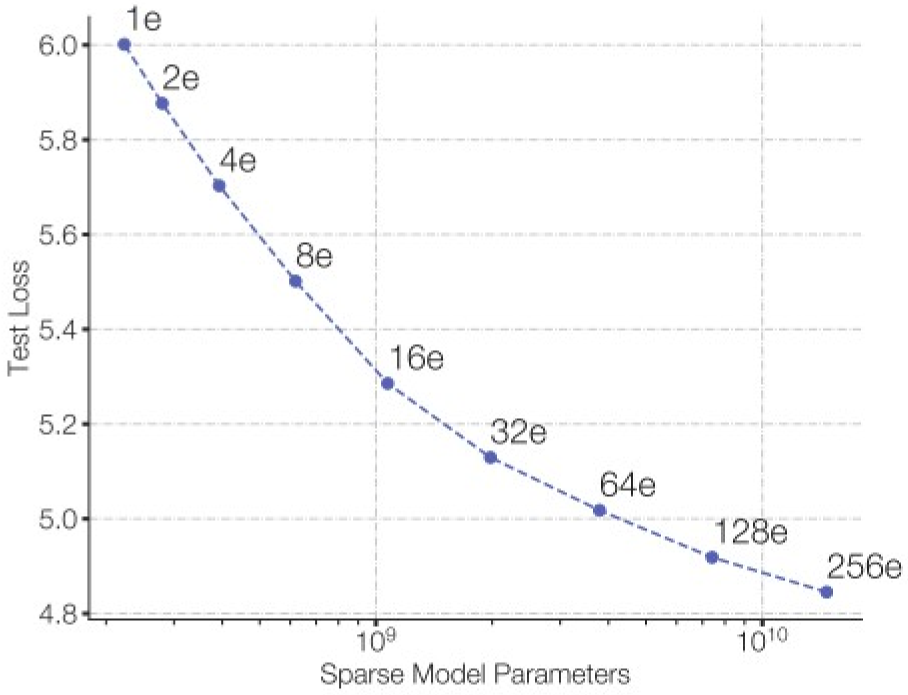
- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.
- **Expert Capacity:** Batch size of each expert. Calculated as
- (tokens_per_batch / num_experts) * capacity_factor
- **Capacity Factor:** Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.

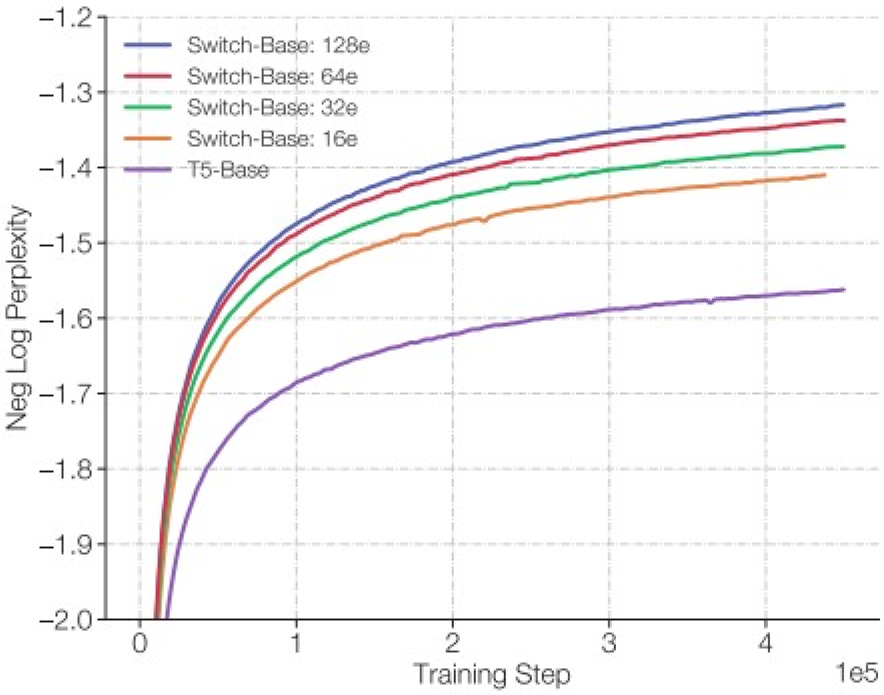


MoE Variant: Switch Transformer

Switch: https://arxiv.org/pdf/2101.03961.pdf

MLM pre-training objective [BERT-like]

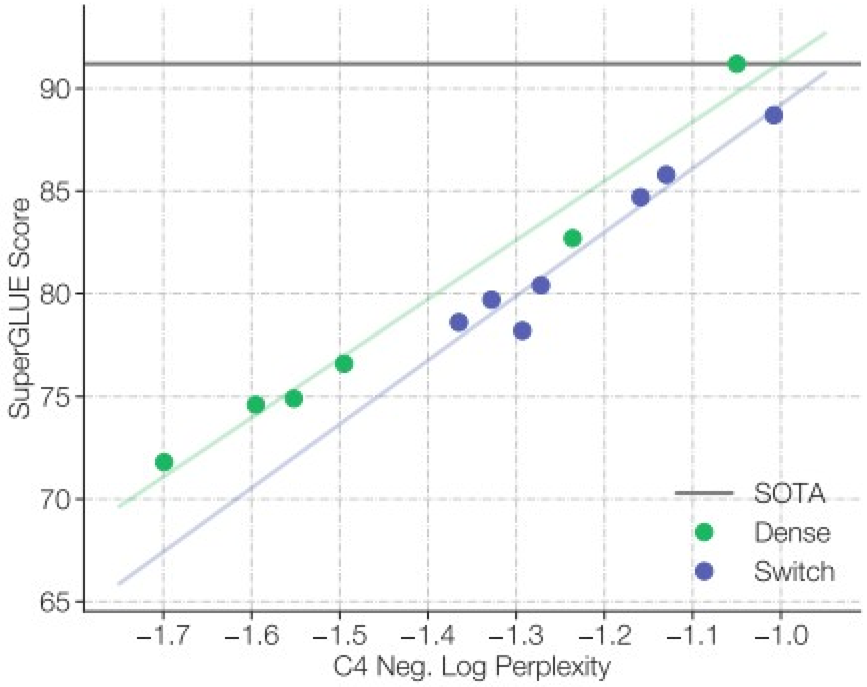


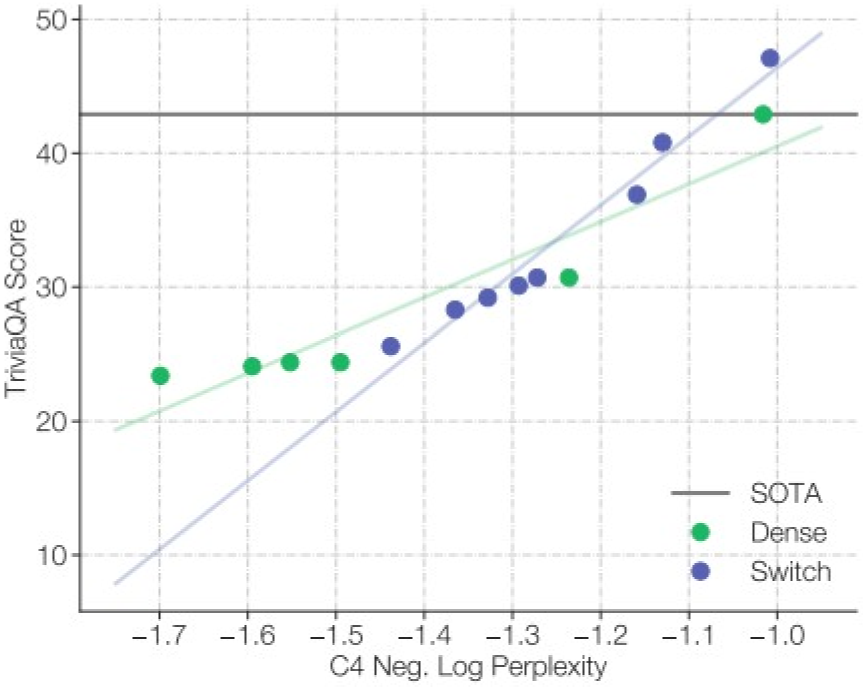


MoE Variant: Switch Transformer

Switch: https://arxiv.org/pdf/2101.03961.pdf

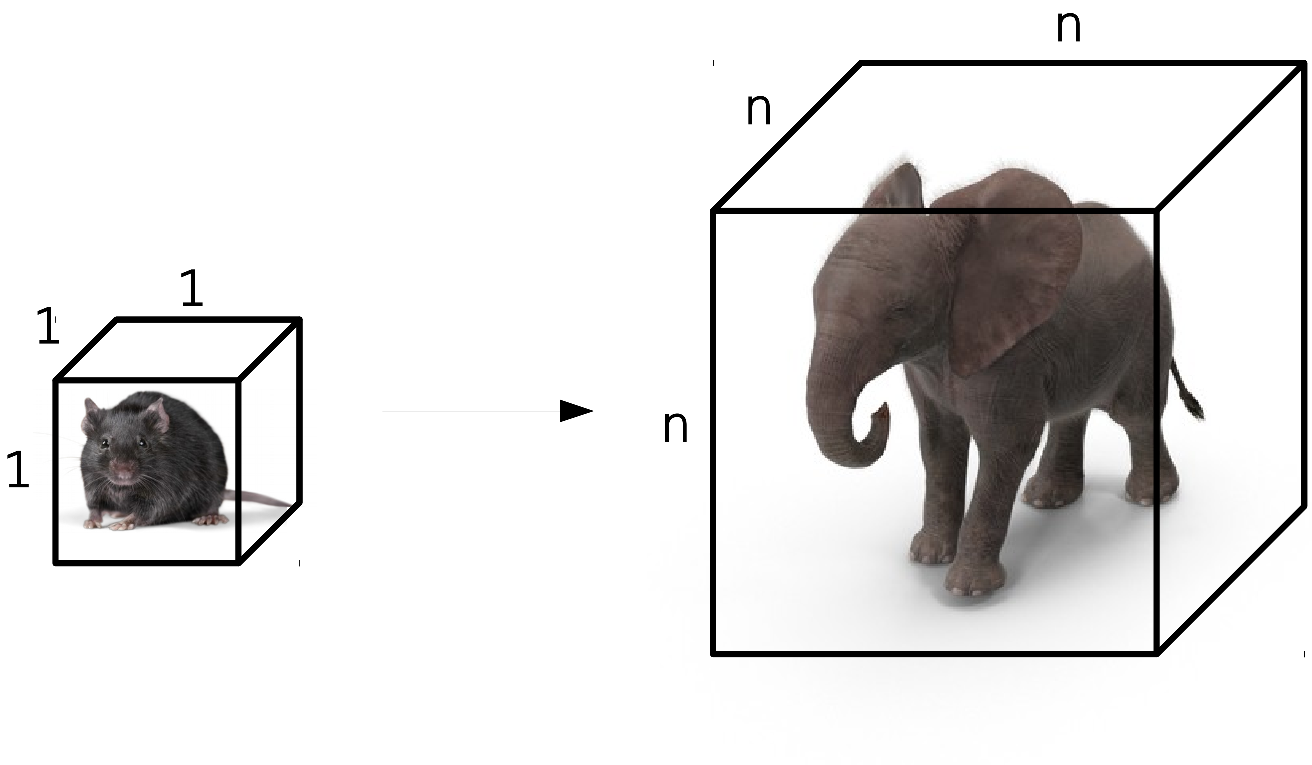
Pre-training vs downstream quality





The square-cube law of deep learning

Explainer: https://www.youtube.com/watch?v=f7KSfjv4Oq0

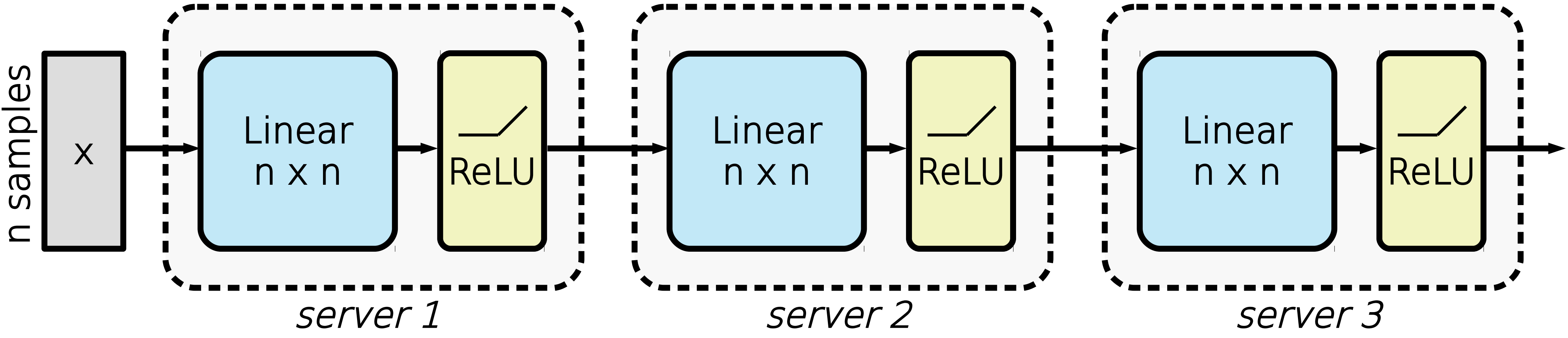


Surface area $O(n^2)$

Volume $O(n^3)$

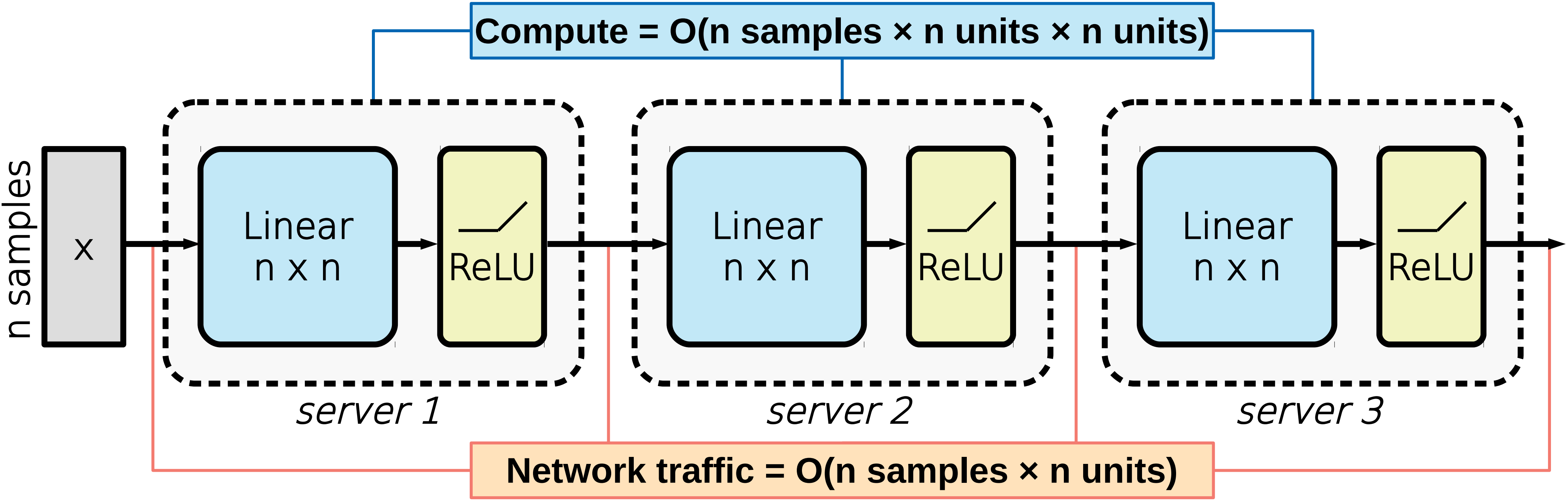
The square-cube law of deep learning

Explainer: https://www.youtube.com/watch?v=f7KSfjv4Oq0



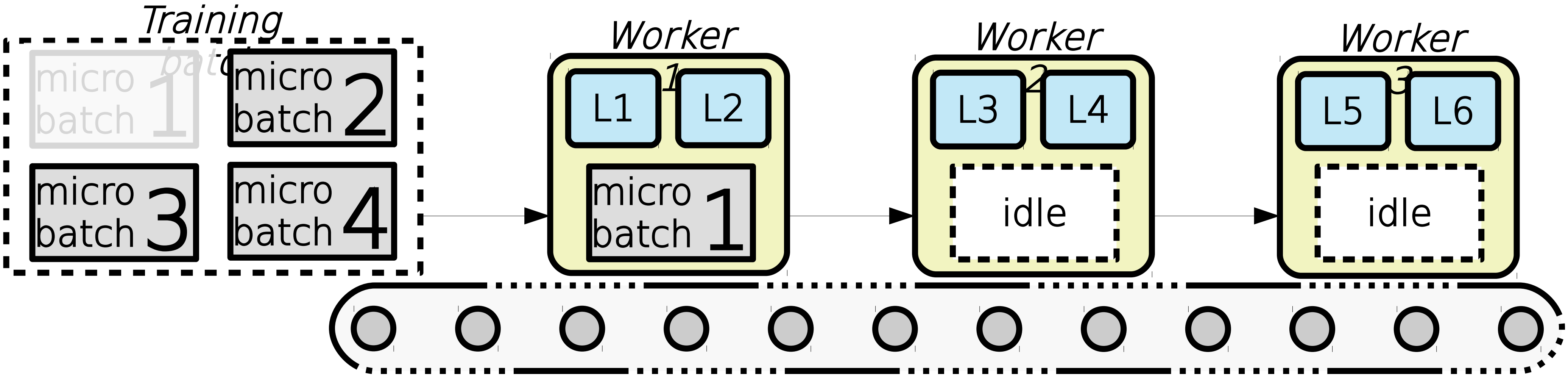
The square-cube law of deep learning

Explainer: https://www.youtube.com/watch?v=f7KSfjv4Oq0



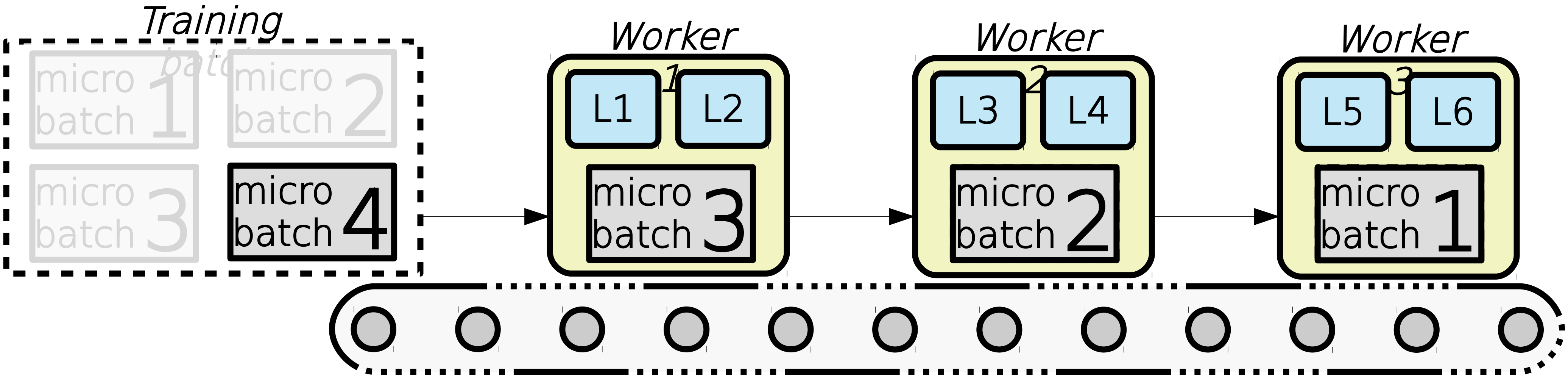
The square-cube law of deep learning

Explainer: https://www.youtube.com/watch?v=f7KSfjv4Oq0



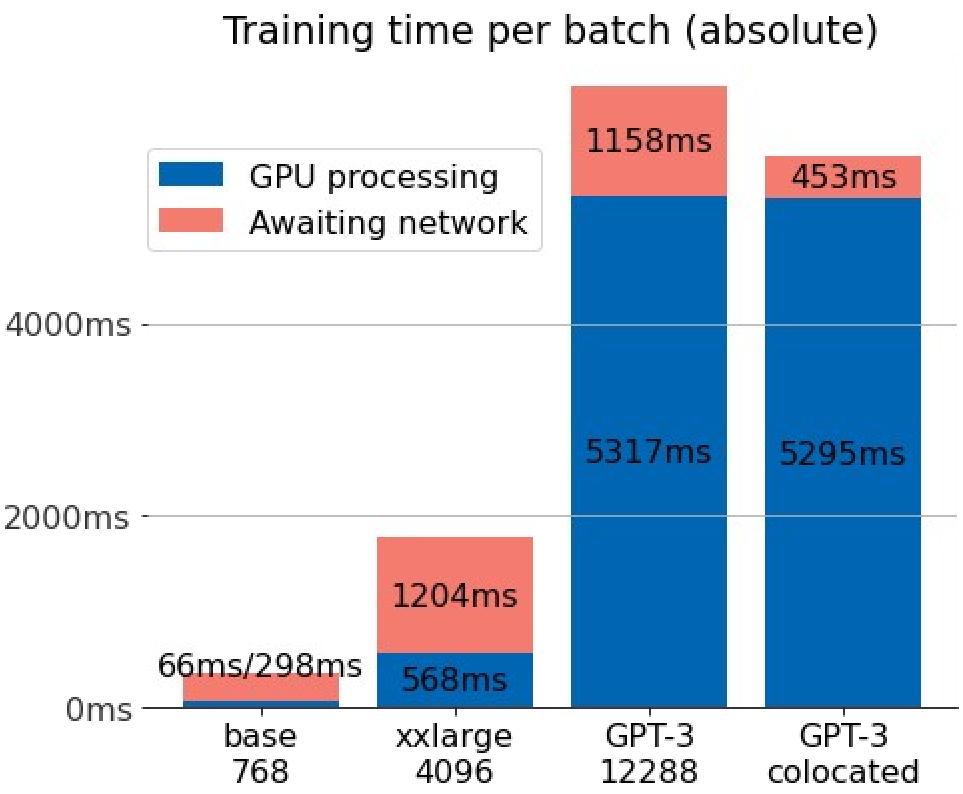
The square-cube law of deep learning

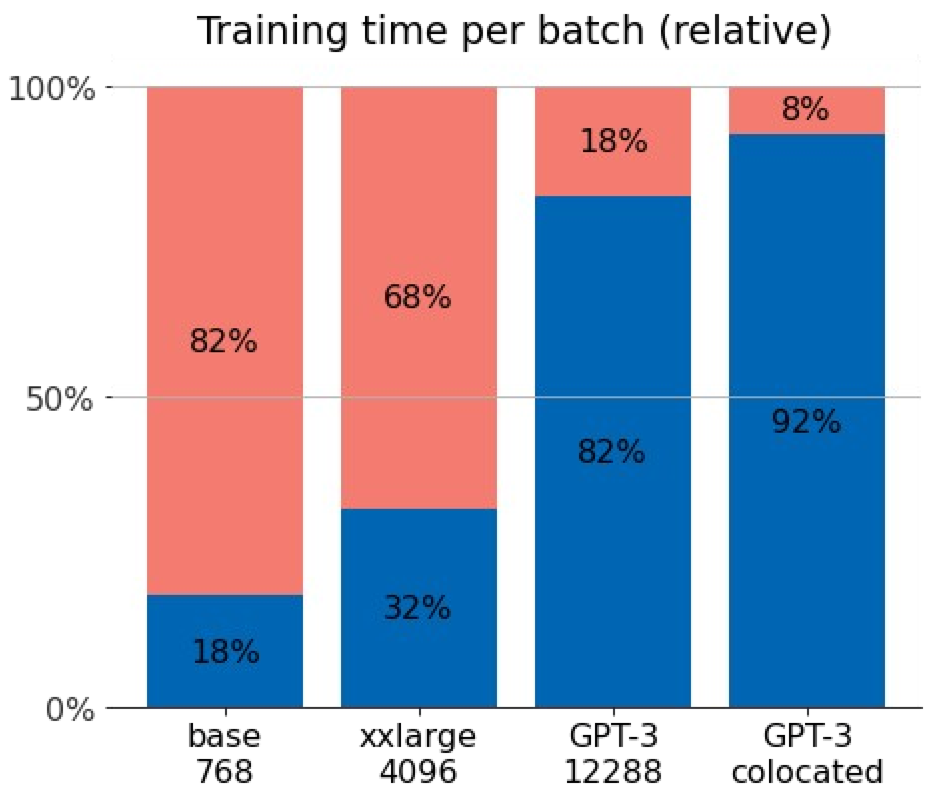
Explainer: https://www.youtube.com/watch?v=f7KSfjv4Oq0



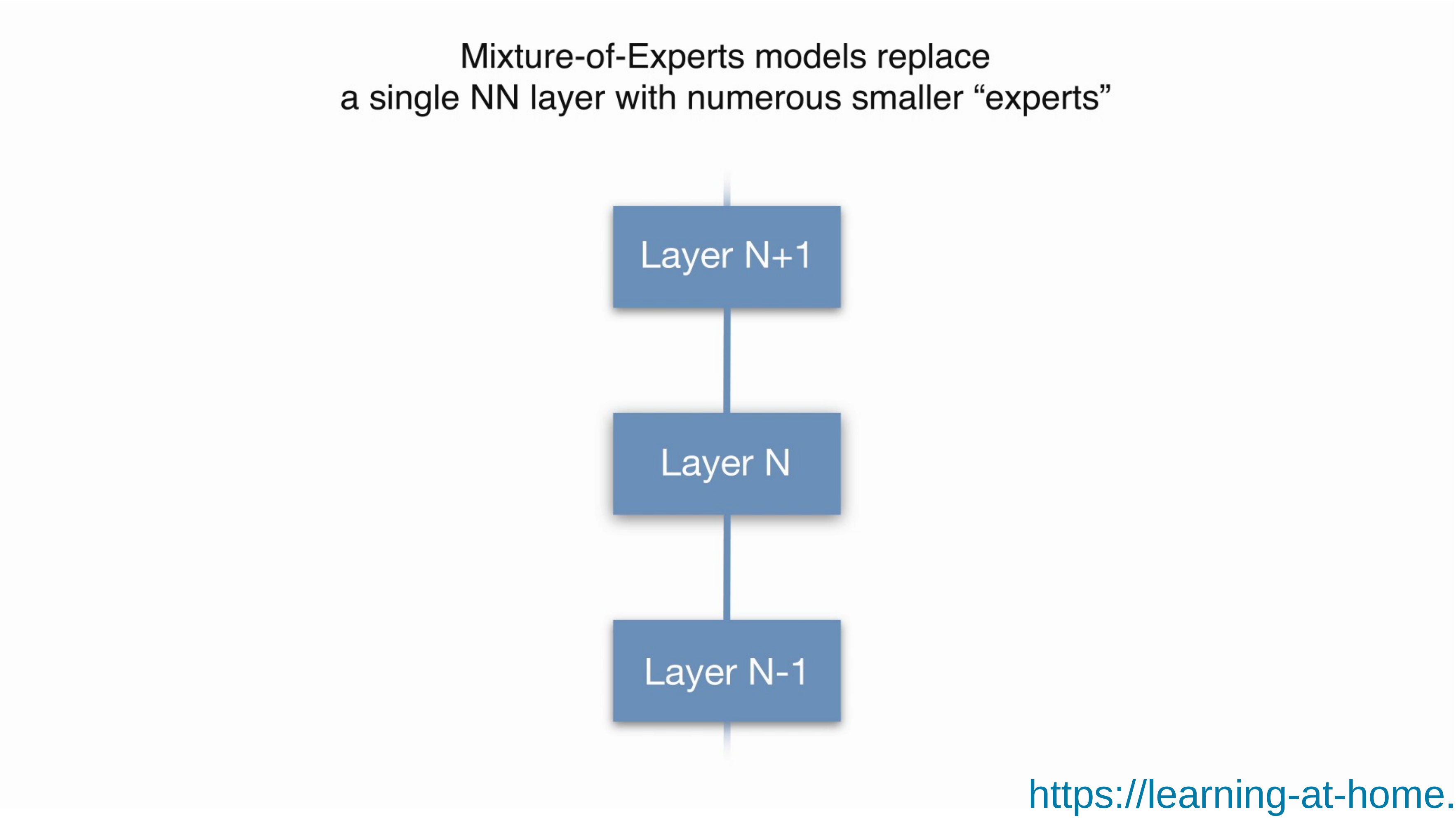
The square-cube law of deep learning

Explainer: https://www.youtube.com/watch?v=f7KSfjv4Oq0





Hivemind



https://learning-at-home.github.io/

</lecture>