Fedor Ratnikov

Parallel & distributed training

2021

MLHEP 2021

NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS

LAMBDA • HSE

Yandex

SCHOOL OF DATA ANALYSIS

EPFL

SIT Schaffhausen Institute of Technology

Large problems need large models

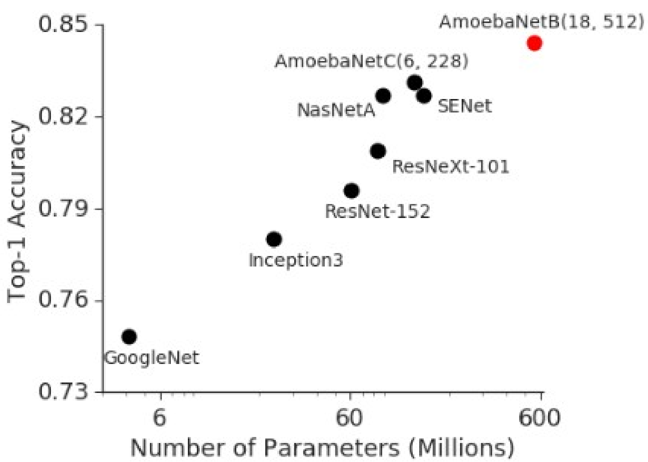
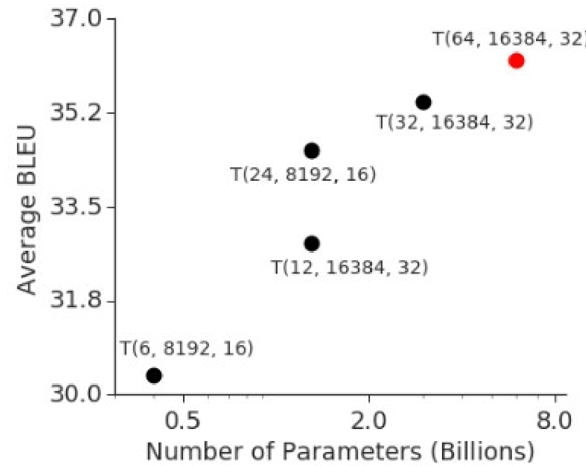


Image Classification
ImageNet

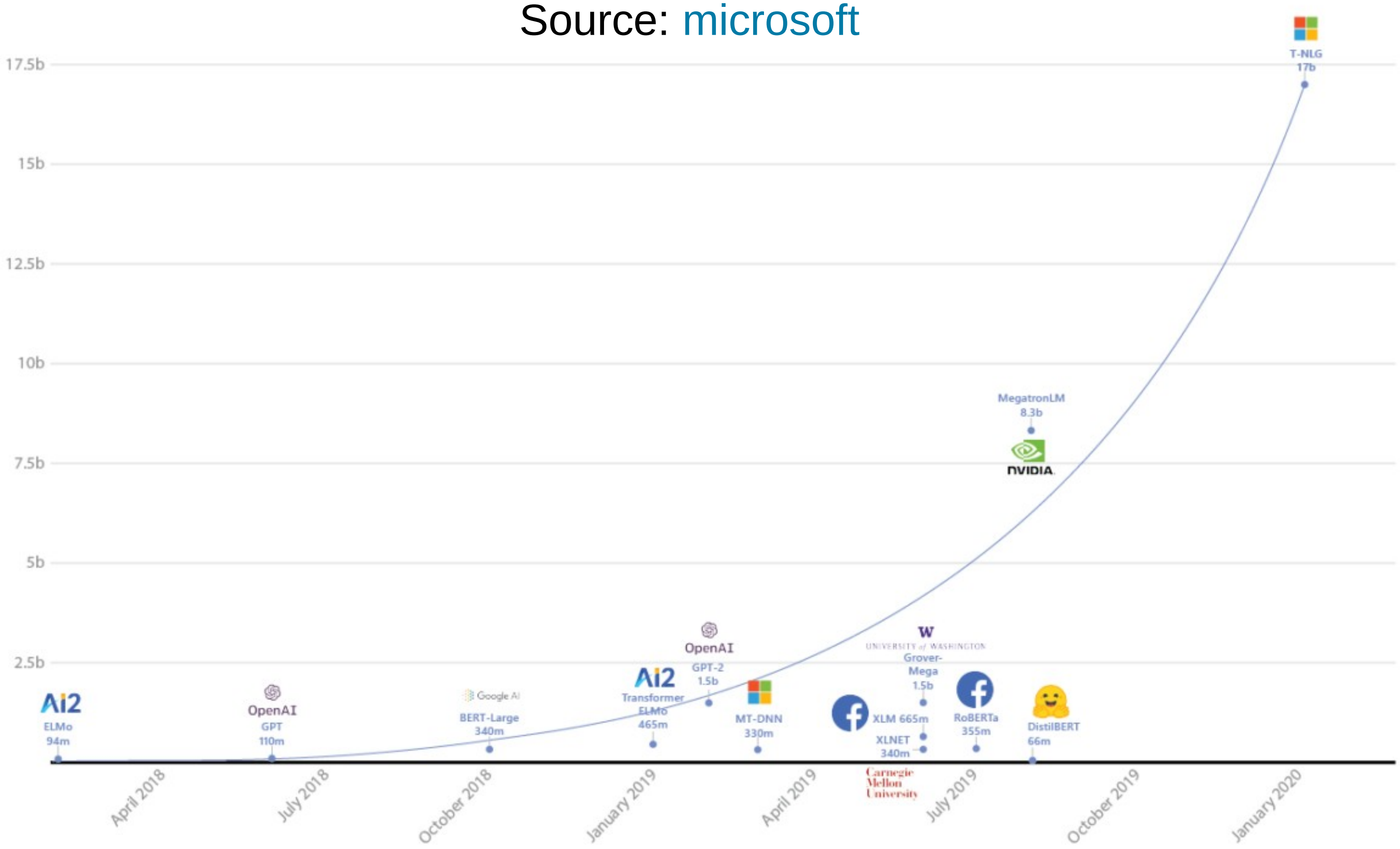


Machine Translation
average over WMT

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

The transformer curve

Source: microsoft



Machine Learning Supertasks

Image classification – ImageNet, JFT300M

Generative models – ImageNet(biggan), the internet

Language Models – common crawl, BERT / MLM

Machine Translation – multilingual translation

Reinforcement Learning – playstation* & steam :)

* playstation for RL: https://arxiv.org/abs/1912.06101

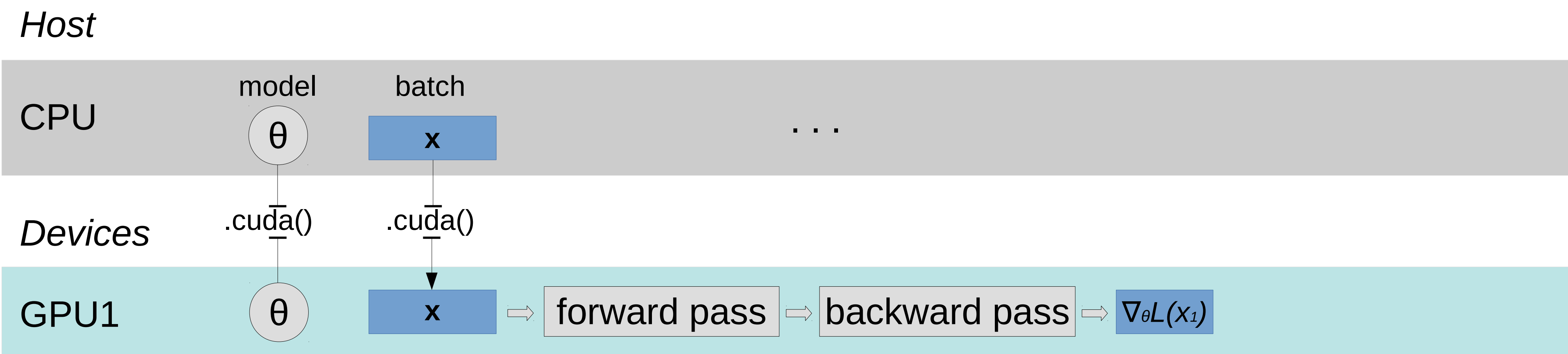
Meanwhile, exabytes of YouTube videos lay dormant across the web, waiting for someone who can make use of them

Data-parallel training (naive)

cs.cmu.edu/~muli/file/parameter_server_osdi14.pdf

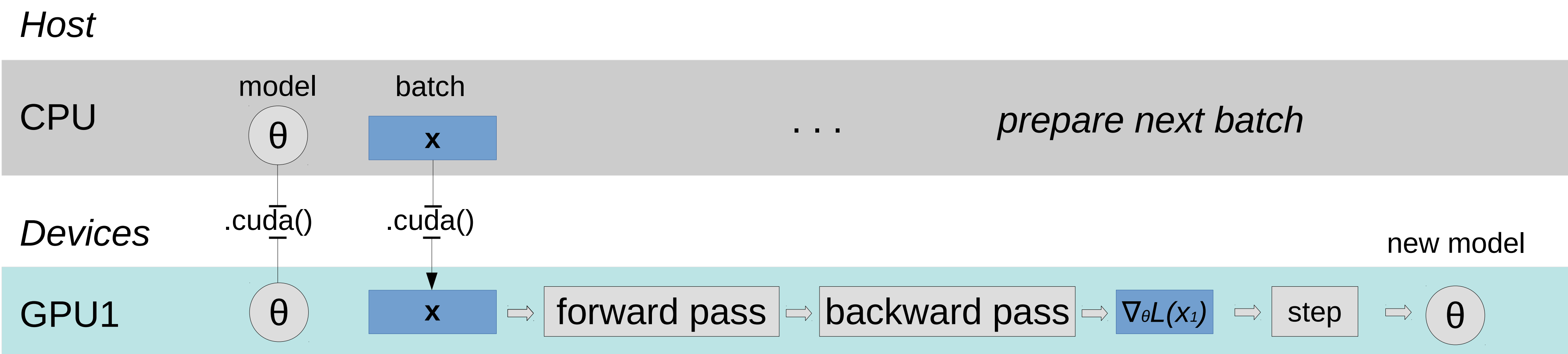
Data-parallel training (naive)

cs.cmu.edu/~muli/file/parameter_server_osdi14.pdf



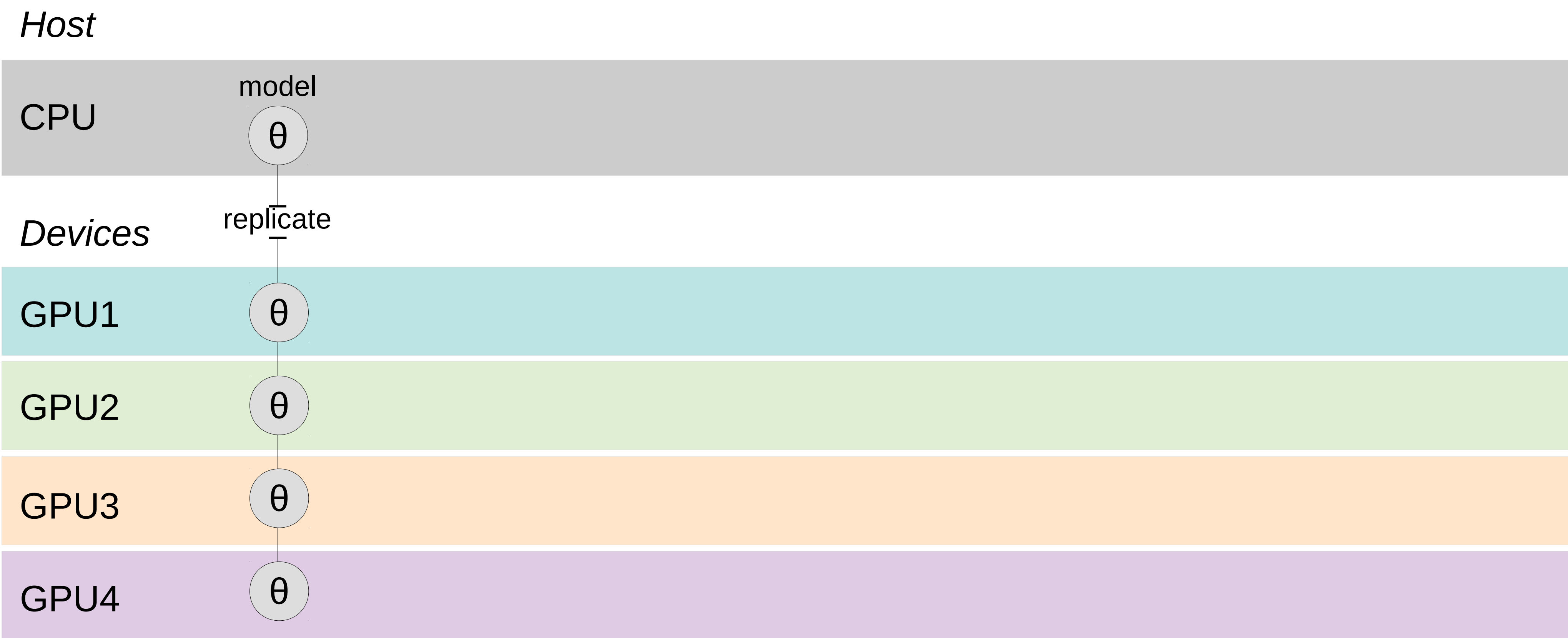
Data-parallel training (naive)

cs.cmu.edu/~muli/file/parameter_server_osdi14.pdf



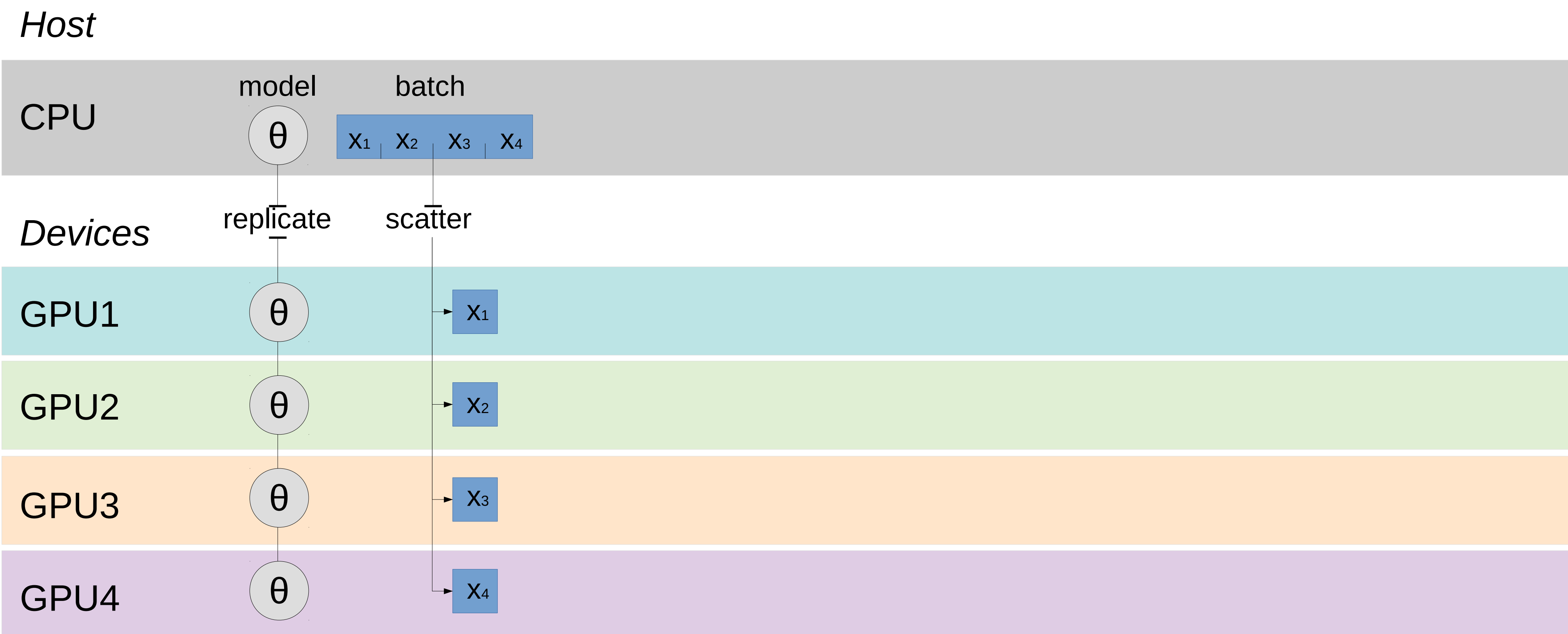
Data-parallel training (naive)

cs.cmu.edu/~muli/file/parameter_server_osdi14.pdf



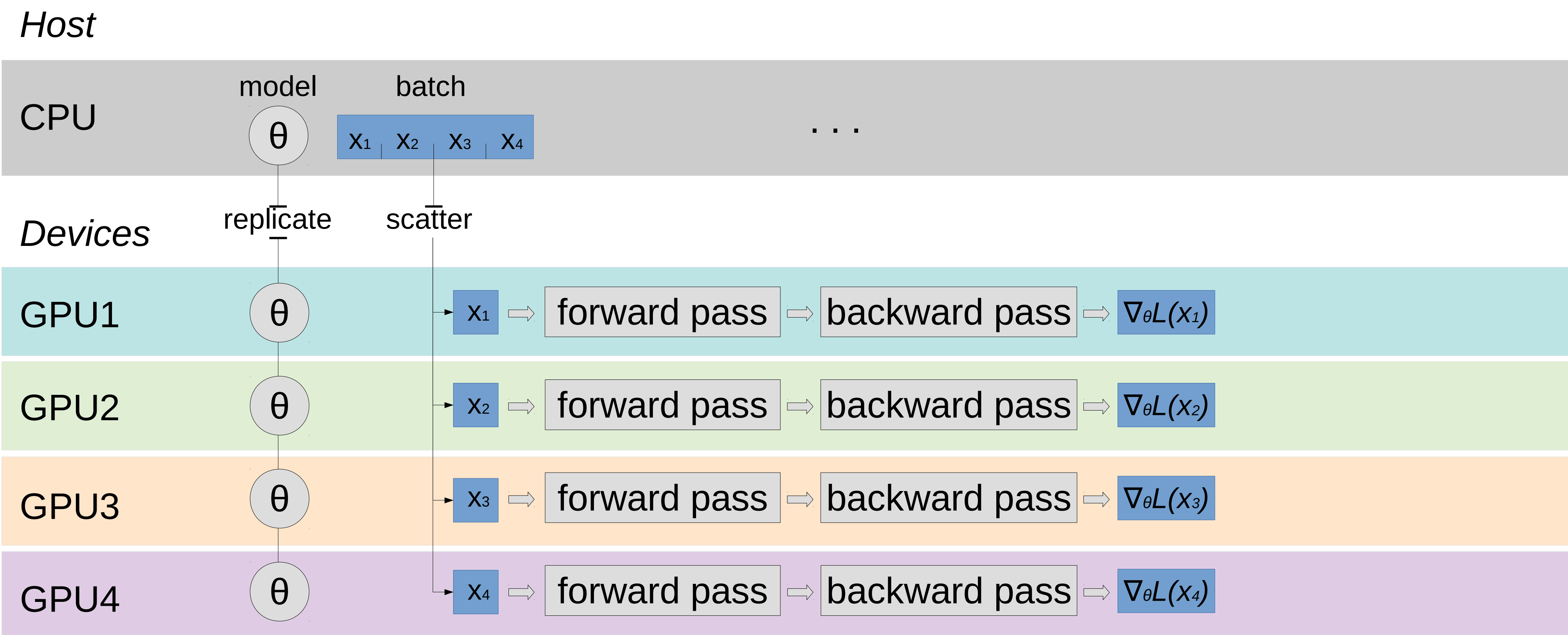
Data-parallel training (naive)

cs.cmu.edu/~muli/file/parameter_server_osdi14.pdf



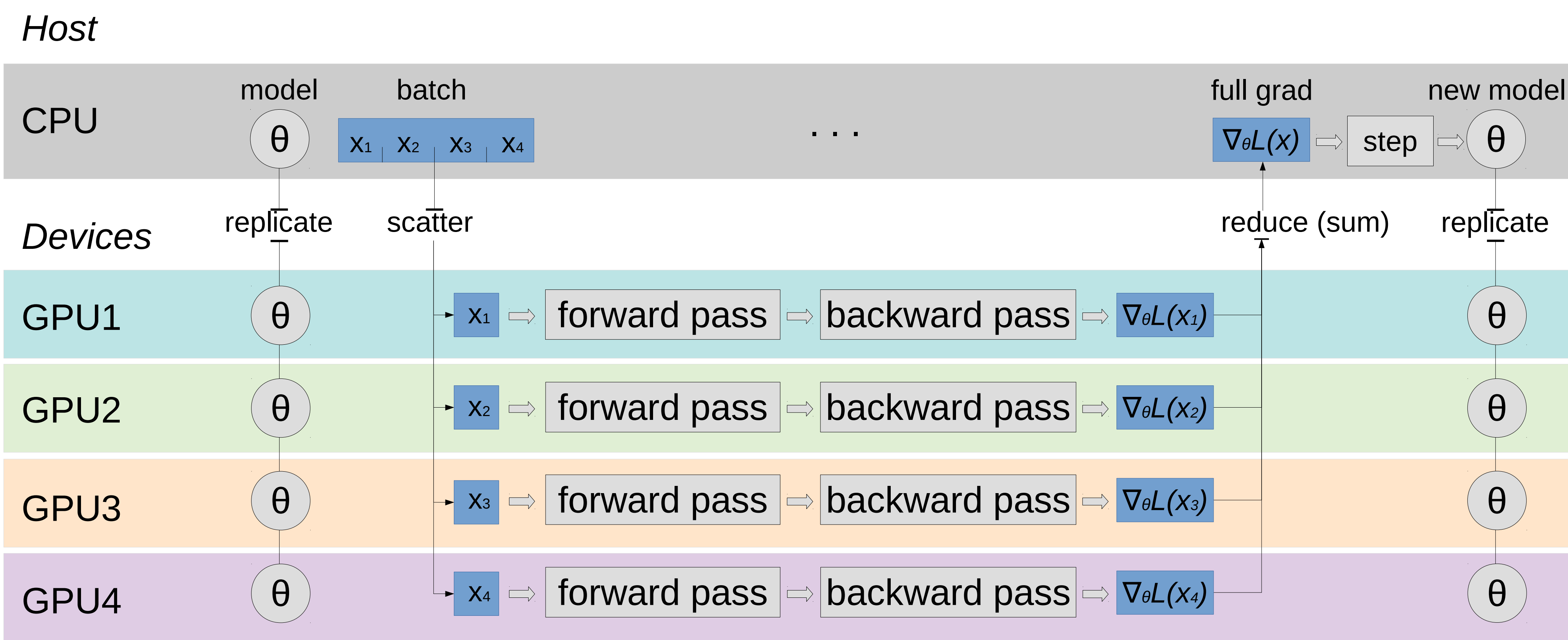
Data-parallel training (naive)

cs.cmu.edu/~muli/file/parameter_server_osdi14.pdf



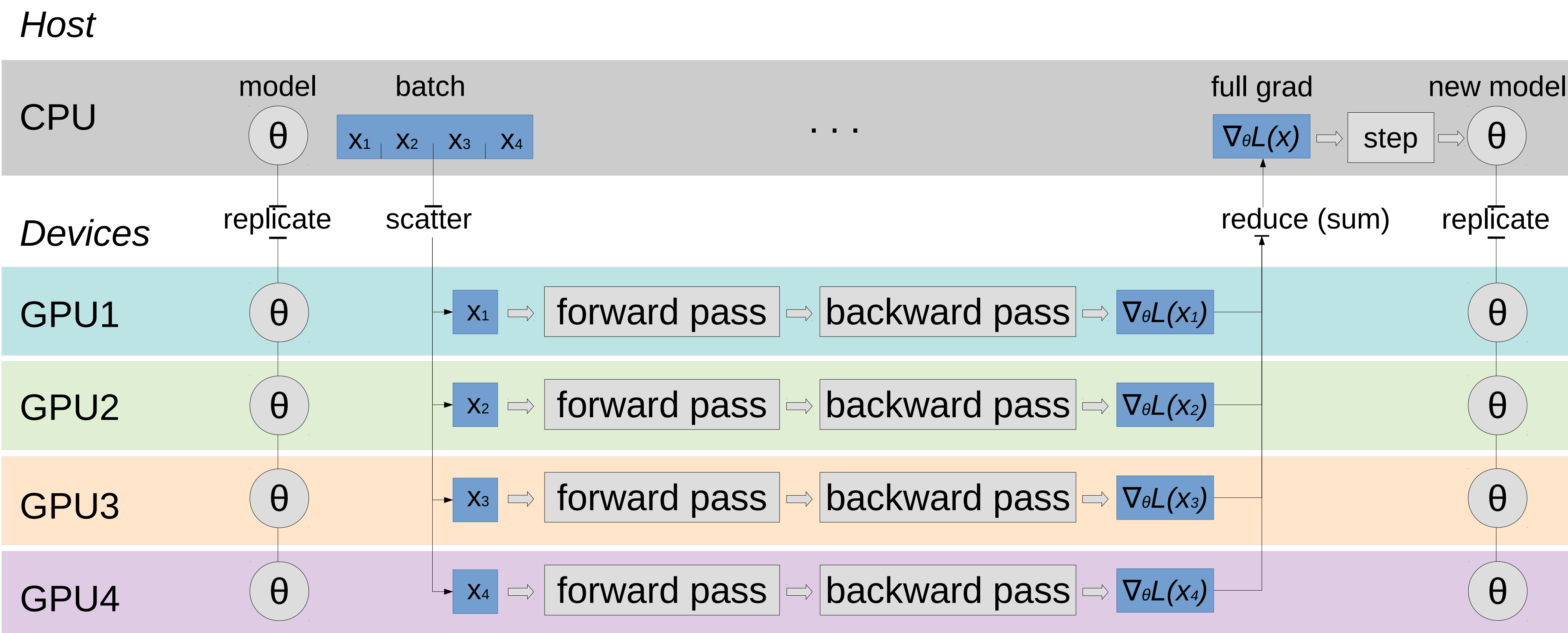
Data-parallel training (naive)

cs.cmu.edu/~muli/file/parameter_server_osdi14.pdf



Data-parallel training (naive)

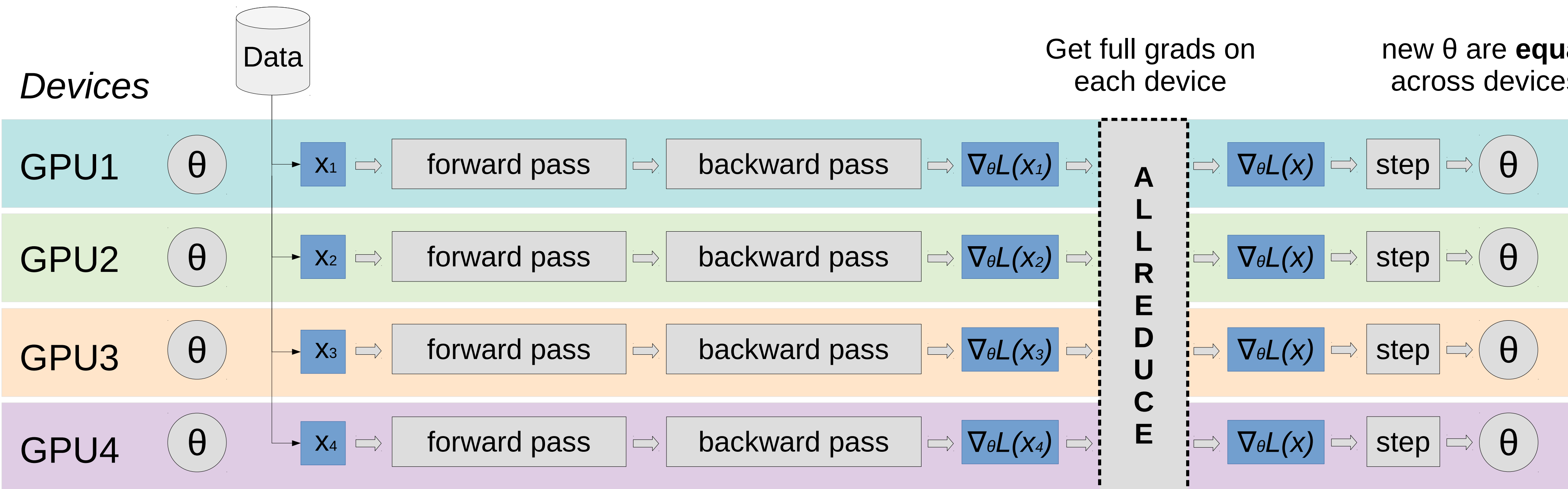
cs.cmu.edu/~muli/file/parameter_server_osdi14.pdf



Advanced data parallel

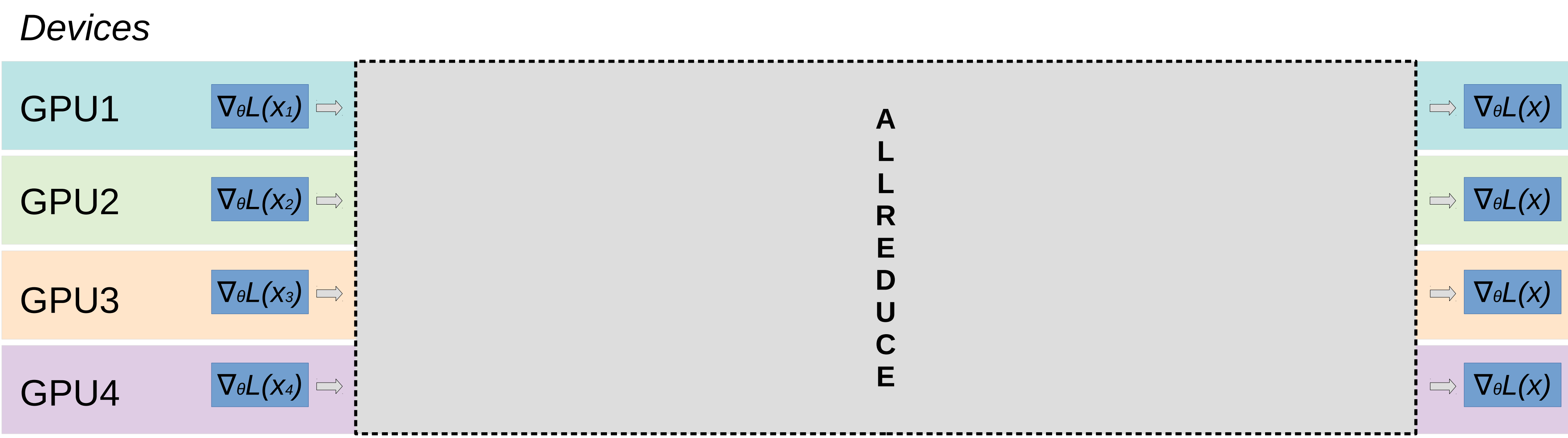
arxiv.org/abs/1706.02677

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



Faster allreduce

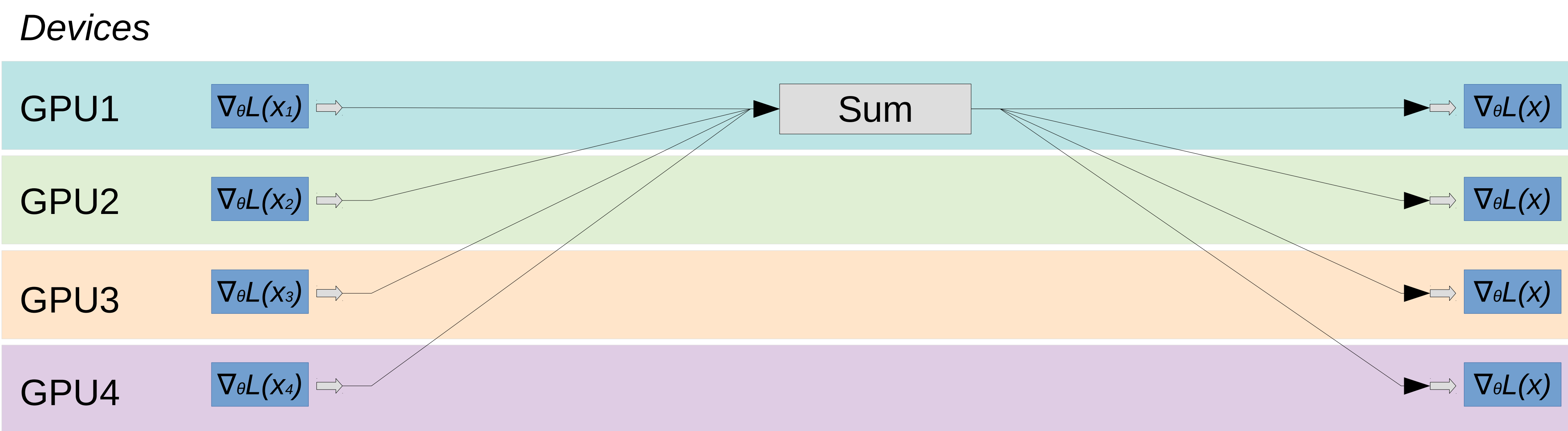
Input: each device has its its own vector
Output: each device gets a sum of all vectors



Faster allreduce

Input: each device has its its own vector
Output: each device gets a sum of all vectors

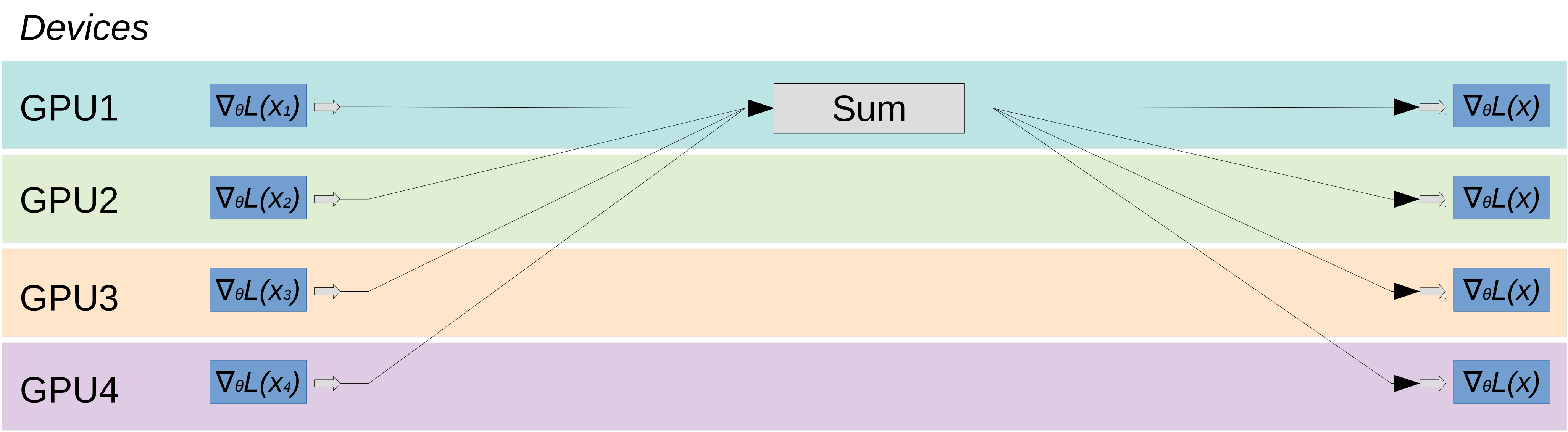
Naive implementation



Faster allreduce

Input: each device has its its own vector
Output: each device gets a sum of all vectors

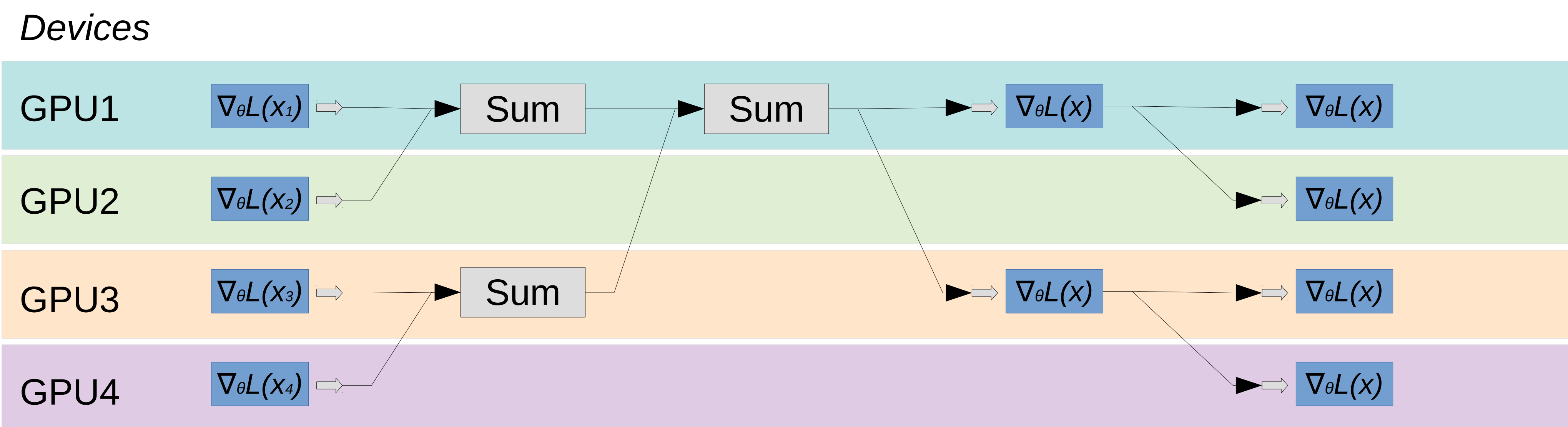
Q: Can we do better?



Faster allreduce

Input: each device has its its own vector
Output: each device gets a sum of all vectors

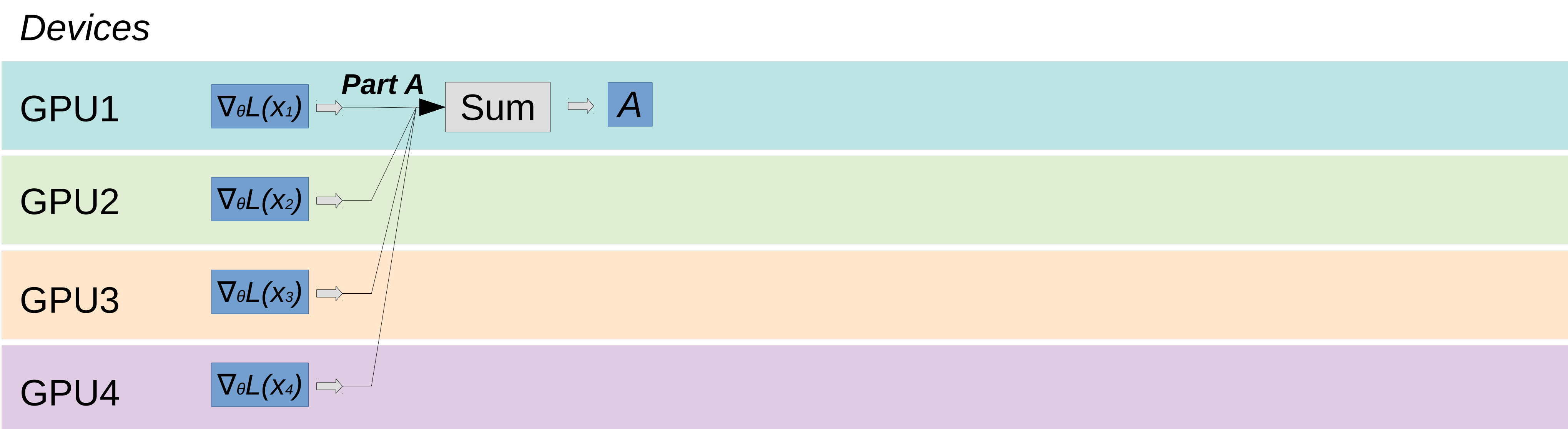
Tree-allreduce



Faster allreduce

Input: each device has its its own vector
Output: each device gets a sum of all vectors

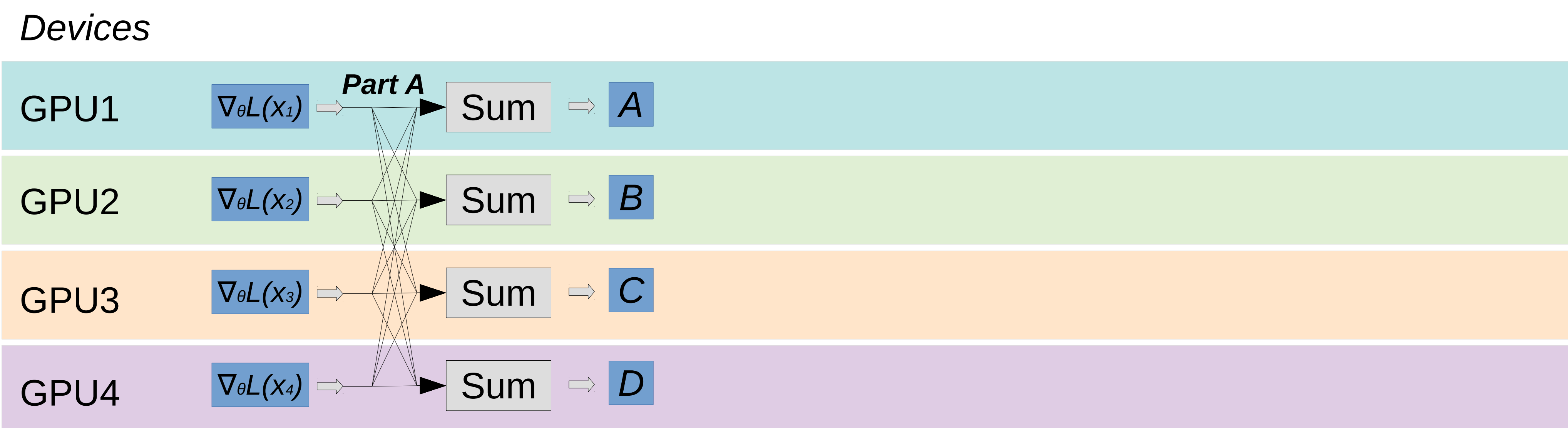
Split data into chunks (ABCD)



Faster allreduce

Input: each device has its its own vector
Output: each device gets a sum of all vectors

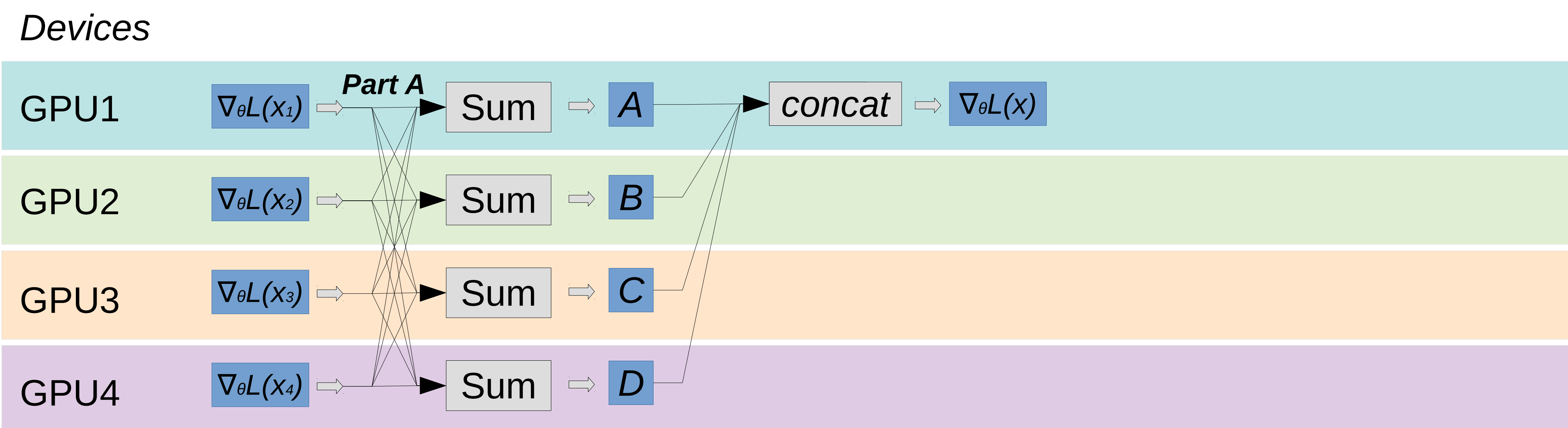
Split data into chunks (ABCD)



Faster allreduce

Input: each device has its its own vector
Output: each device gets a sum of all vectors

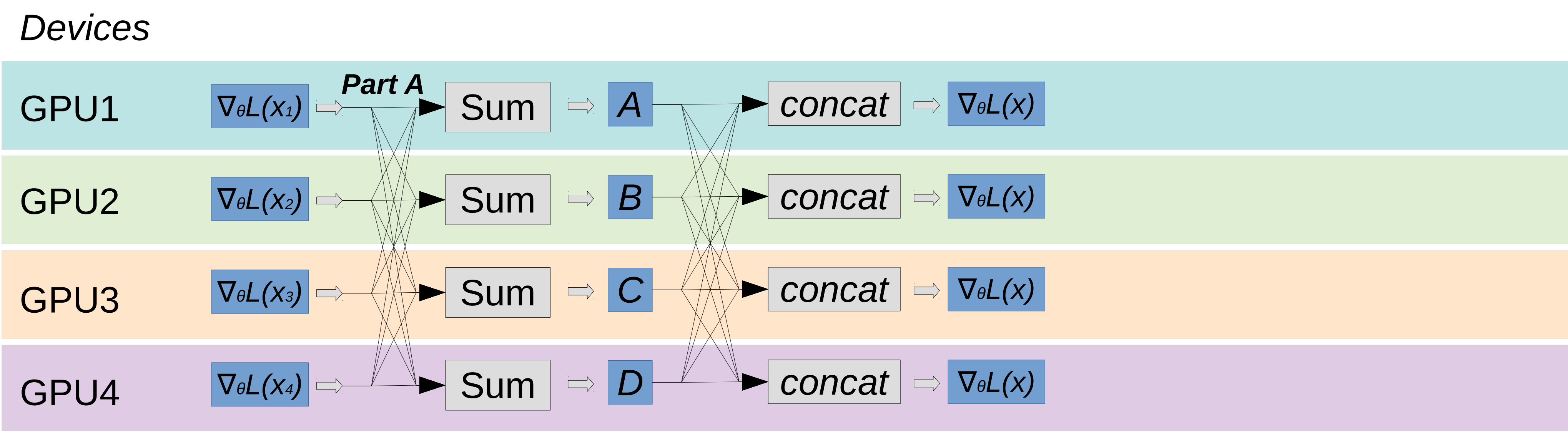
Split data into chunks (ABCD)



Faster allreduce

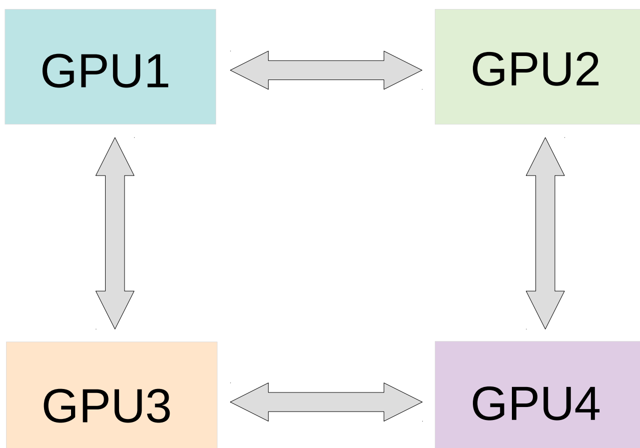
Input: each device has its its own vector
Output: each device gets a sum of all vectors

Split data into chunks (ABCD)



Ring all-reduce

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



Ring topology

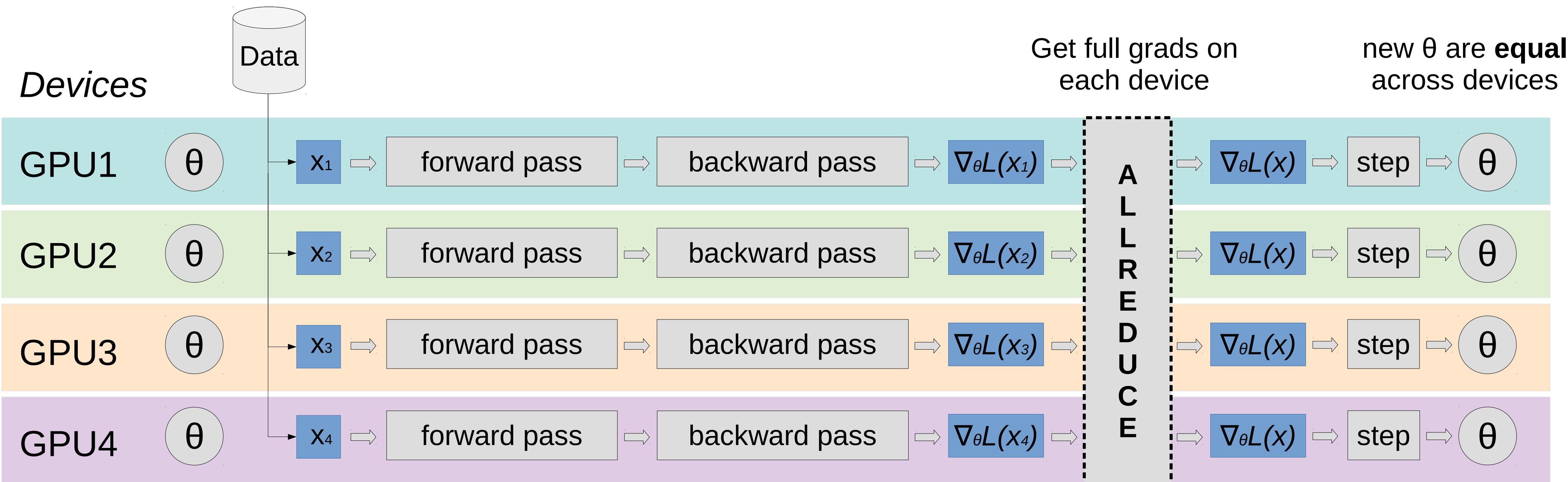
Image: graphcore ipu server

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

Advanced data parallel

arxiv.org/abs/1706.02677

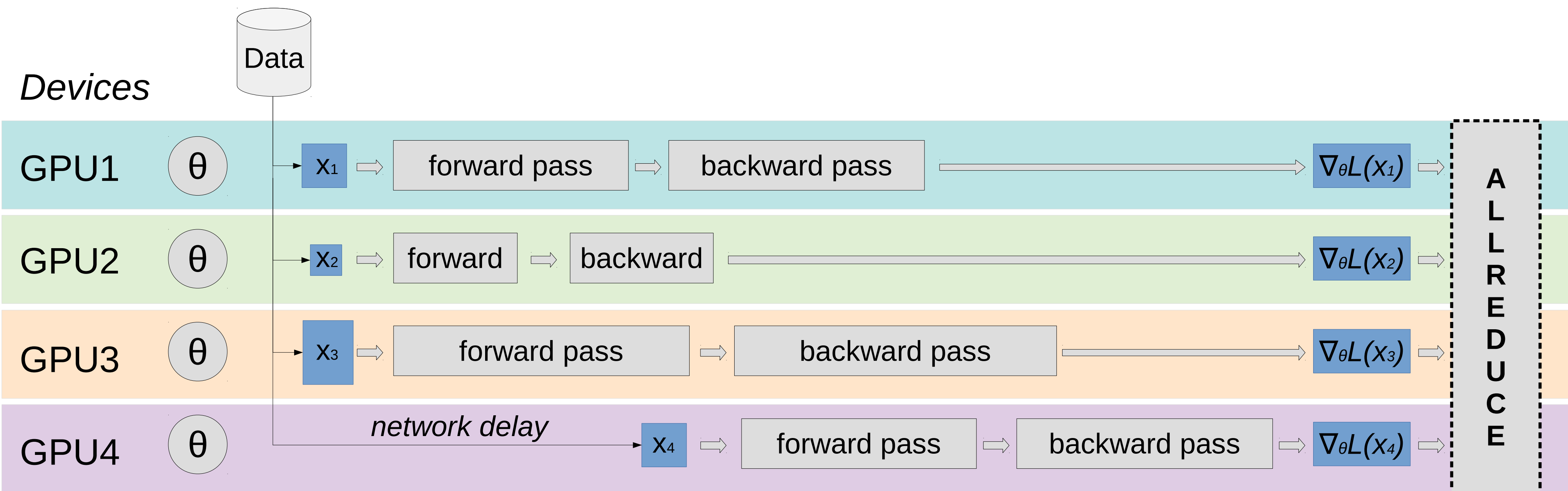
Idea: get rid of the host, each gpu runs its own computation
Q: why will weights be equal after such step?



Advanced data parallel vs reality

arxiv.org/abs/1706.02677

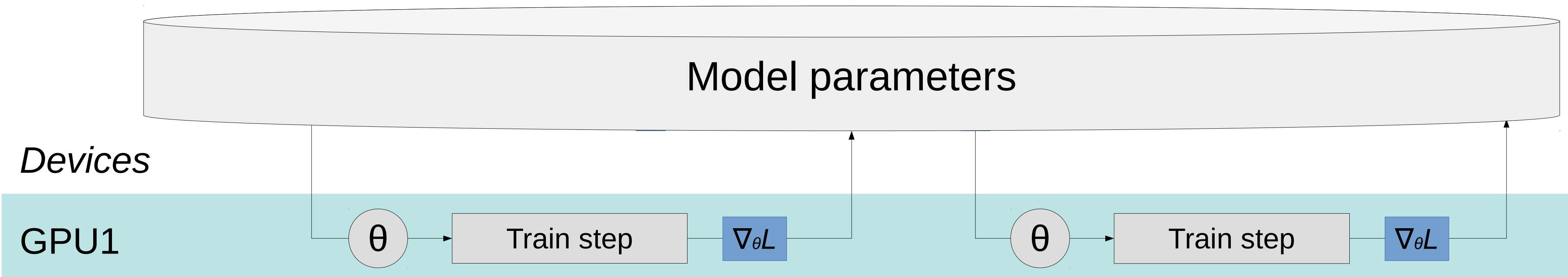
Each gpu has different processing time & delays
Q: can we improve device utilization?



Asynchronous data-parallel training

HOGWILD! arxiv.org/abs/1106.5730

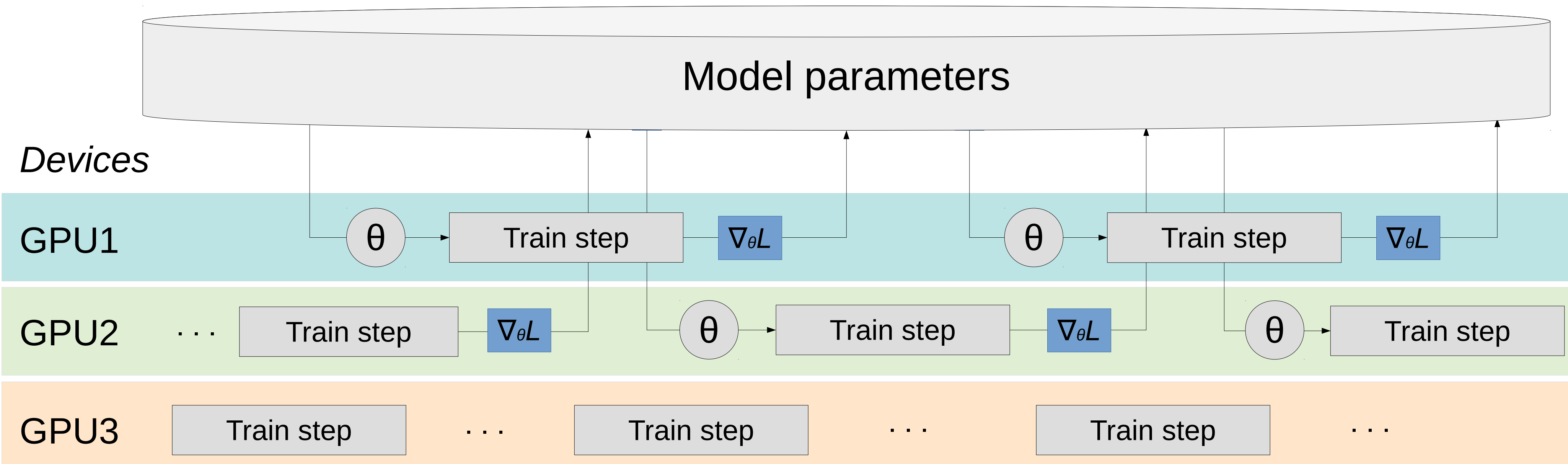
Idea: remove synchronization step alltogether, use parameter server



Asynchronous data-parallel training

HOGWILD! arxiv.org/abs/1106.5730

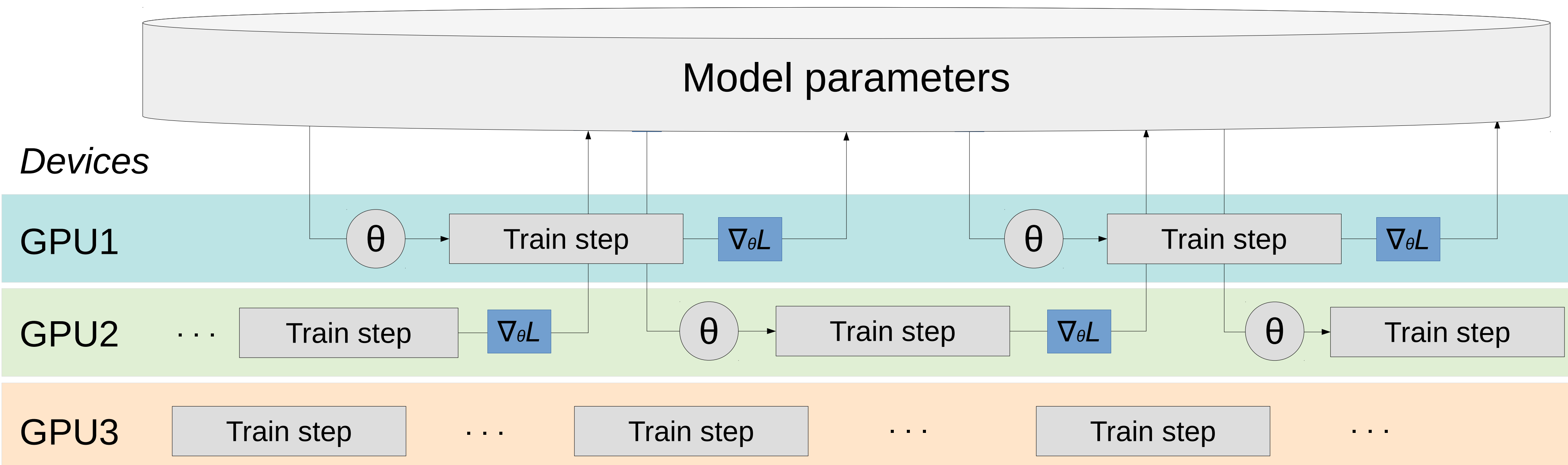
Idea: remove synchronization step alltogether, use parameter server



Asynchronous data-parallel training

HOGWILD! arxiv.org/abs/1106.5730

Idea: remove synchronization step alltogether, use parameter server

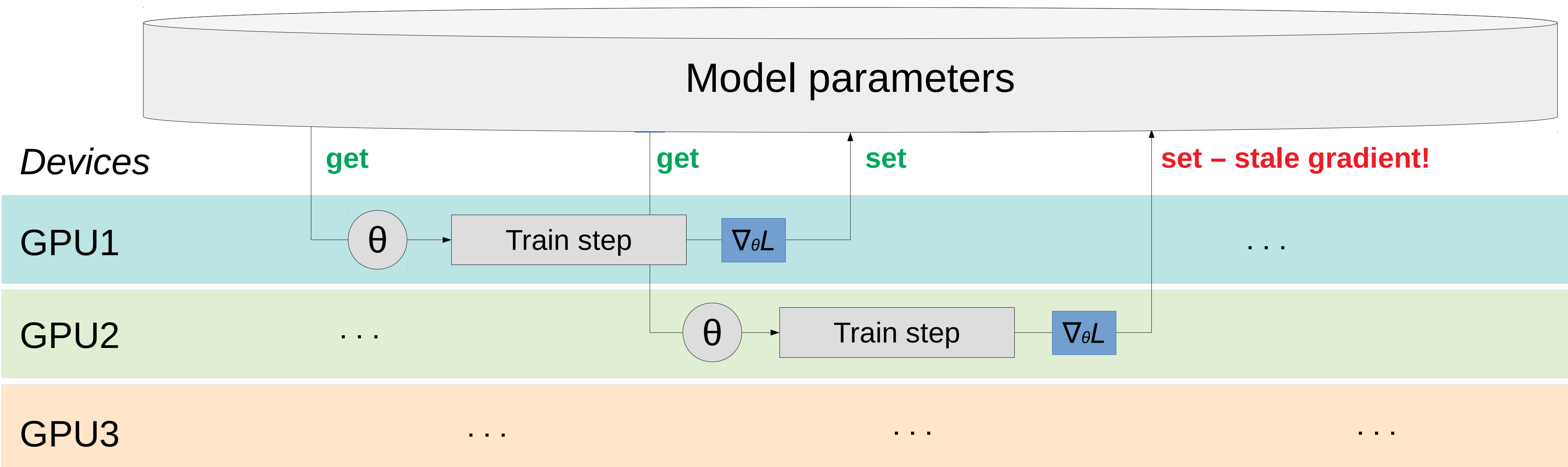


Q: have we lost anything by going asynchronous?

Asynchronous data-parallel training

HOGWILD! arxiv.org/abs/1106.5730

Idea: remove synchronization step alltogether, use parameter server



Correction for staleness: arxiv.org/abs/1511.05950 & many others

Data-parallel Reinforcement Learning

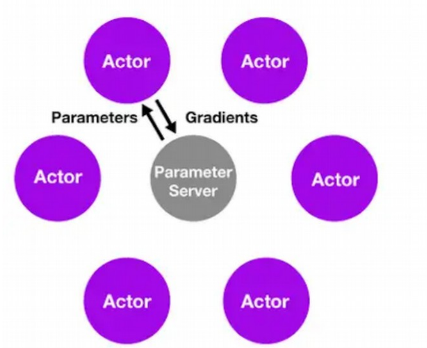
Synchronous data-parallel: A. Stooke & P. Abbeel, 2018 tinyurl.com/gtc-parallel-rl

Asynchronous data-parallel:

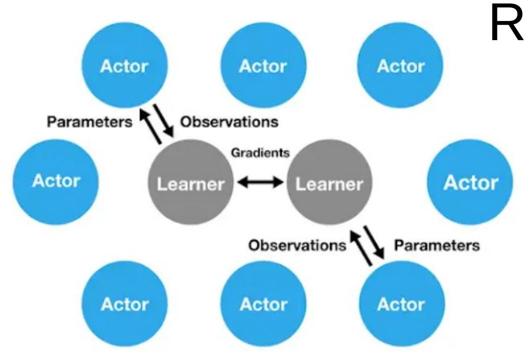
Asynchronous methods for deep RL: arxiv.org/abs/1602.01783

Distributed asynchronous data-parallel:

IMPALA: arxiv.org/abs/1802.01561
R2D2: openreview.net/forum?id=r1lyTjAqYX
SEED RL: arxiv.org/abs/1910.06591



A3C



IMPALA - Multiple Learners

More:

(english) https://youtu.be/kOy49NqZeqI
(russian) https://youtu.be/wswbMkT55mI

</Data-parallel>

+ easy to implement
+ can scale to 100s of gpus
+ can be fault-tolerant
- model must fit in 1 gpu
- large batches aren’t always good for generalization

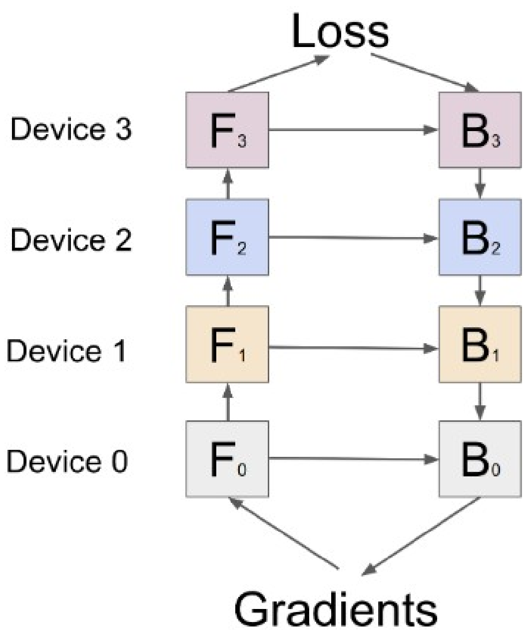
- 2-4 GPUs & no time – naive data parallel tinyurl.com/torch-data-parallel
- 4+ GPUs or multiple hosts – horovod (allreduce) github.com/horovod/horovod
  - High-level distributed pytorch (allreduce): tinyurl.com/distributed-dp
- Somewhat faulty GPU/network: synchronous data parallel + drop stragglers
- Very faulty or uneven resources: asynchronous data parallel (more later)
- Efficient training with large batches: LAMB https://arxiv.org/abs/1904.00962
- Dynamically adding or removing resources: https://tinyurl.com/torch-elastic

Model-parallel training

Q: What if a model is larger than GPU?

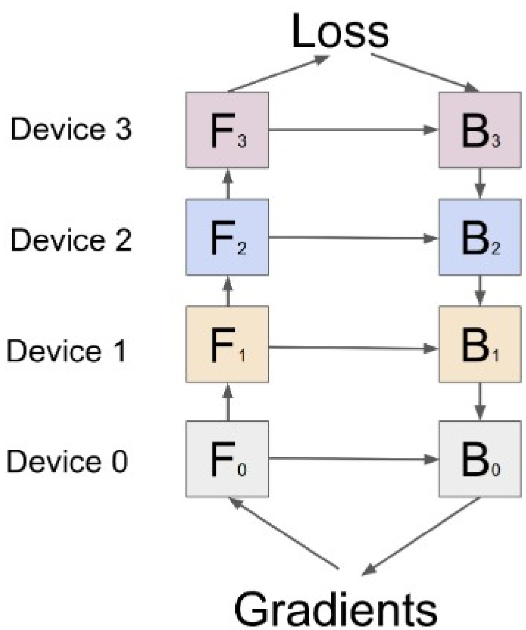
Model-parallel training

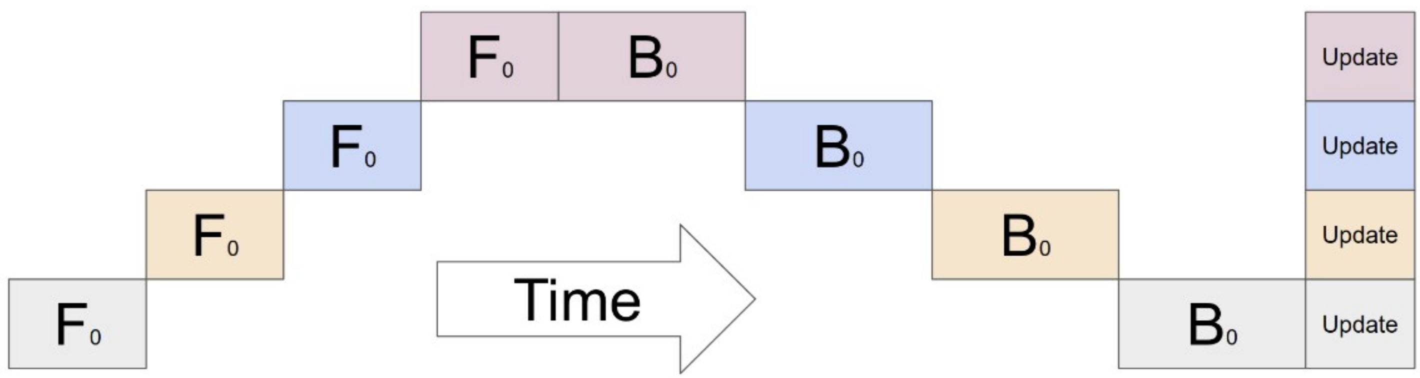
Q: What if a model is larger than GPU?



Model-parallel training

Q: What if a model is larger than GPU?





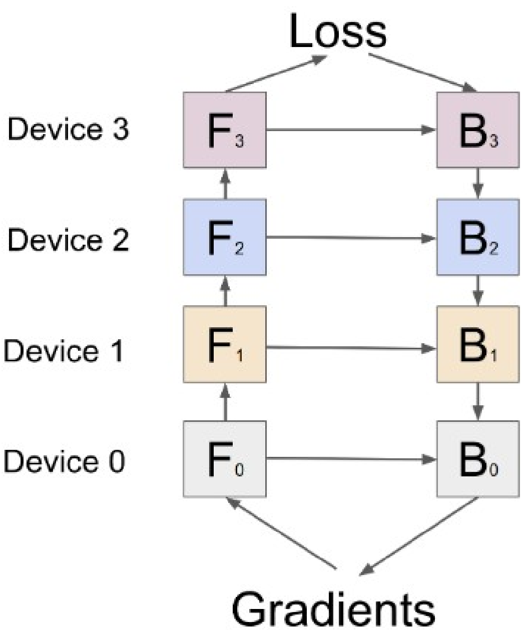
model size: O(N)
throughput: O(1)

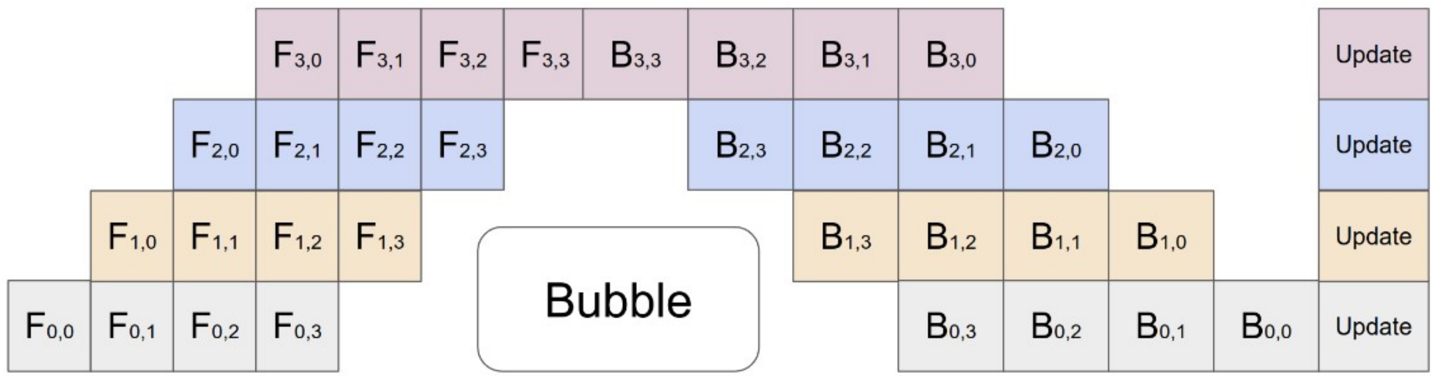
Q: Can we go faster?

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)



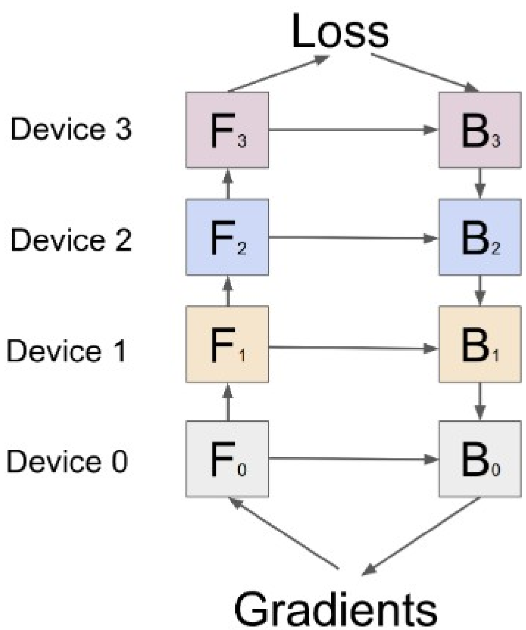


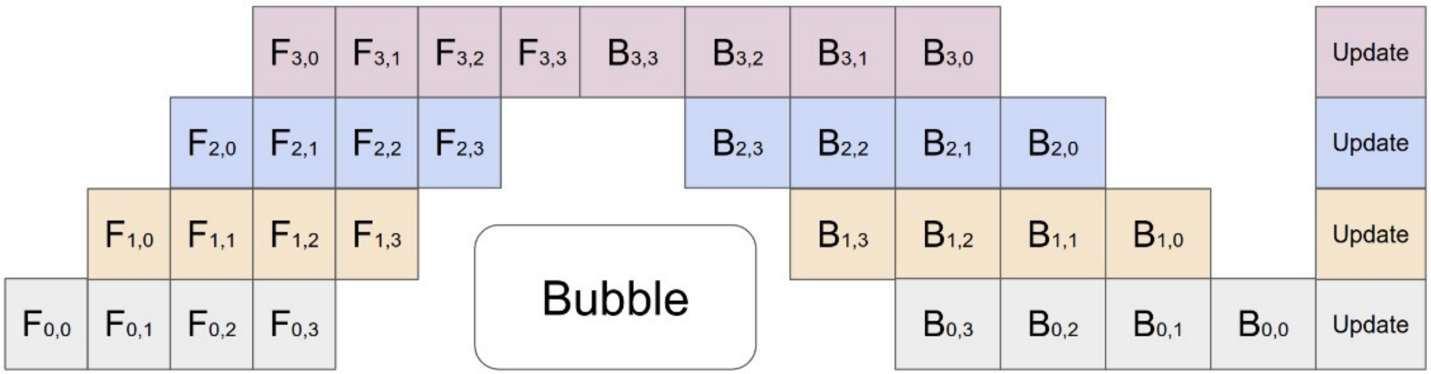
model size: O(n)
throughput: O(n) – with caveats

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?

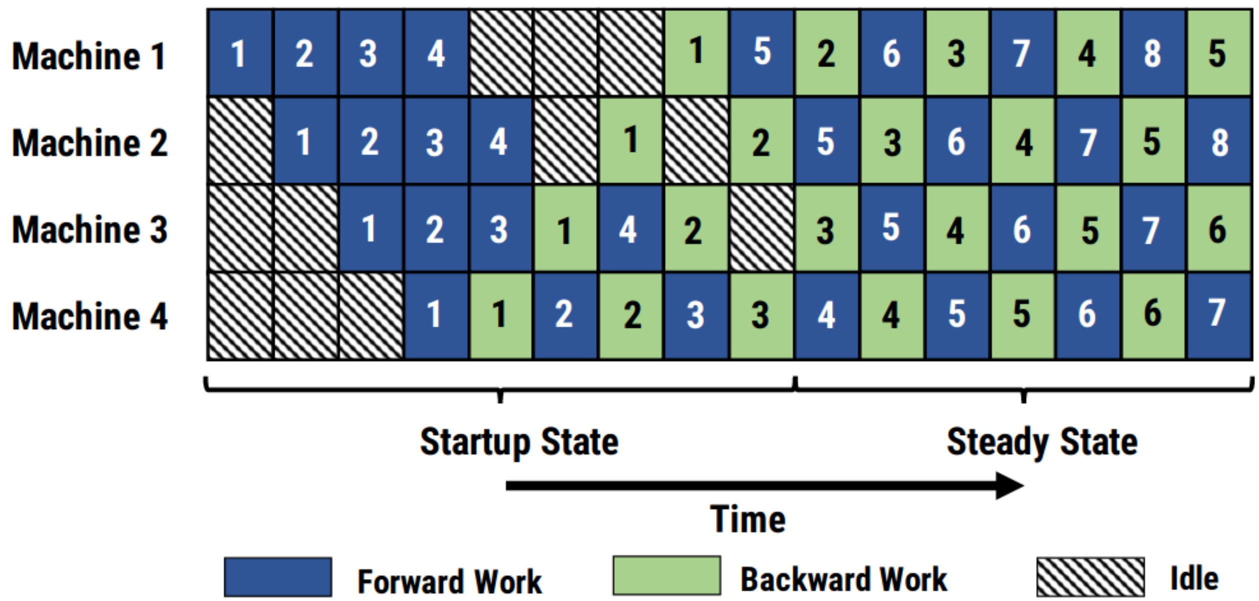
Pipeline-parallel training

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



</Model-parallel>

+ model larger than GPU
+ faster for small
***** typical size: 2-8 gpus
- model partitioning is tricky
- latency is critical, go buy nvlink
except for PipeDream

Tutorials:

- Simple pipelining in PyTorch – tinyurl.com/pytorch-pipelining
- Distributed model-parallel with torch RPC - https://tinyurl.com/torch-rpc
- Advanced but still in active development - github.com/microsoft/DeepSpeed

Case study: DeepSpeed

Source: tinyurl.com/microsoft-deepspeed

ZeRO 4-way data parallel training

Using:

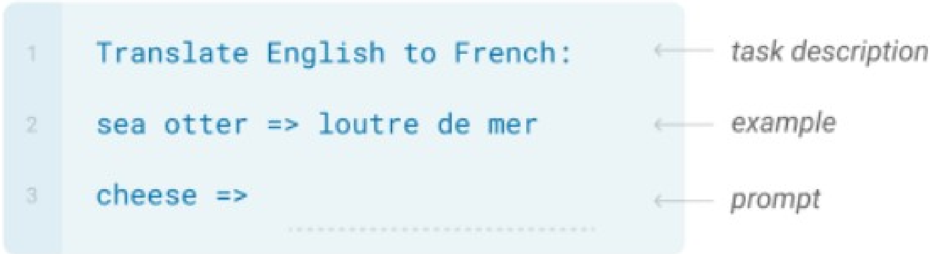
- P_{os} (Optimizer state)
- P_g (Gradient)
- P_p (Parameters)

Case study: GPT-3

Paper: arxiv.org/abs/2005.14165

GPT-3:

- Huge transformer language model, largest = 175B weights
- Can solve many nlp tasks with few or no training examples, see ==>
- Training algorithm: undisclosed
  probably similar to ZeRO
- Dataset: not publicly available
  language & open-source code
- OpenAI being as "open" as ever :)



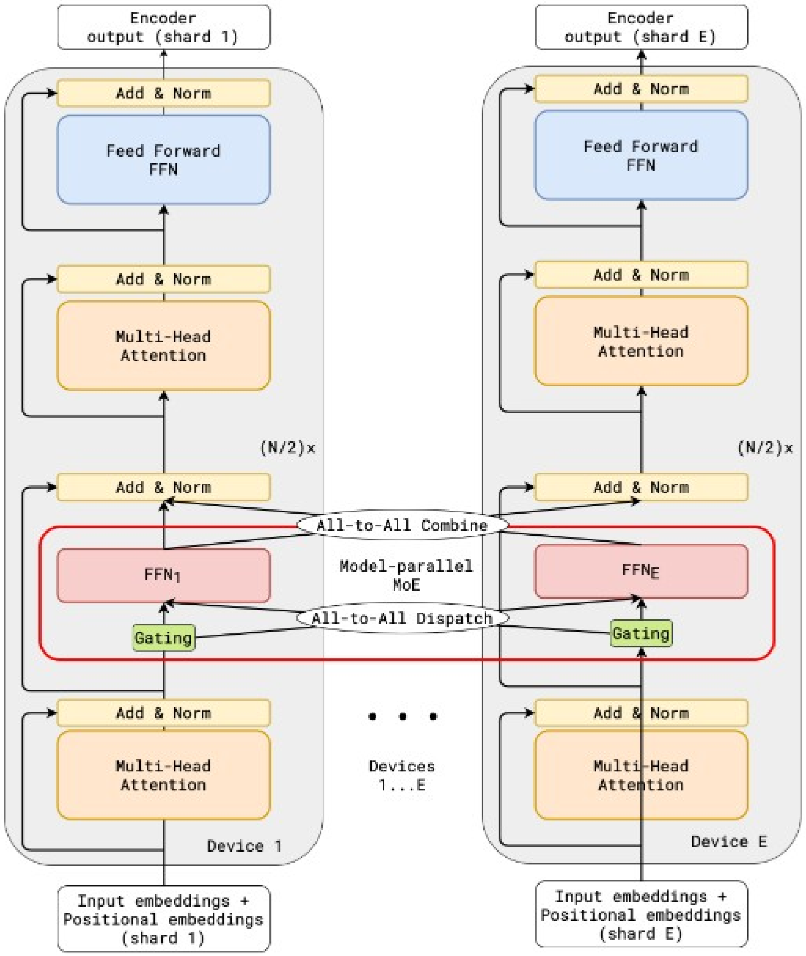

```
Q: When counting, what number comes before 100?
A: Ninety-nine comes before 100.

Q: When counting, what number comes before 123?
A: One hundred and twenty-two comes before 123.

Q: When counting, what number comes before 1000?
A: Nine hundred and ninety-nine comes before 1000.
```

There's always a bigger fish

Gshard paper: arxiv.org/abs/2006.16668
Previous work: arxiv.org/abs/1701.06538

GShard:

- Mixture of experts layers
  - Thousands of small sub-networks
  - Only a few "experts" per input
- Experts are sharded across devices
- Heuristic load-balancing between experts
- 600B parameters. Trained on 2048 TPUs. Because google.



Scaling up vs scaling out

Megatron-LM: arxiv.org/abs/1909.08053
Pricing: phone call to authorized vendor
(anonymized)

Server pod used to train Megatron-LM

- 32x DGX-2H servers ≈ $900,000 each
- 16x Tesla v100-sxm2 sxm3 per node
- High-end interconnect (mellanox / huawei)
- $25±3 million just for the hardware

Scaling up vs scaling out

Megatron-LM: arxiv.org/abs/1909.08053
Pricing: phone call to authorized vendor
(anonymized)

Server pod used to train Megatron-LM

- 32x DGX-2H servers ≈ $900,000 each
- 16x Tesla v100-sxm2 sxm3 per node
- High-end interconnect (mellanox / huawei)
- $25±3 million just for the hardware

Give $25M to gamers:

- 10,000 high-end desktops, $2500 each
- GPUs: mostly 2080Super / 2080Ti
- Home internet connection ↕ 100mbps

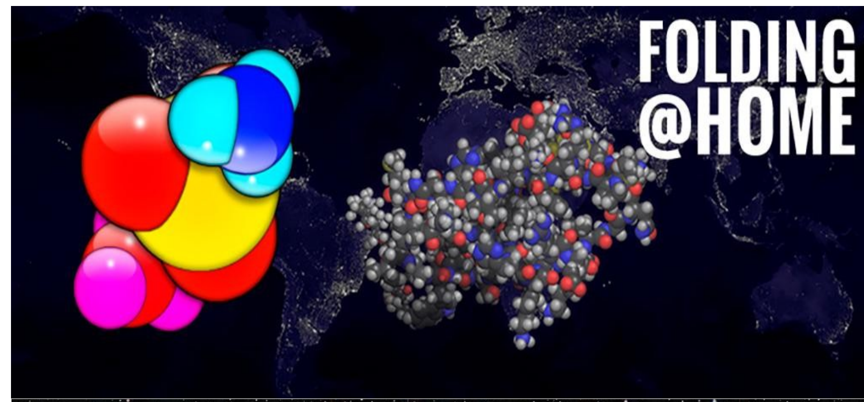
About 10x more flop/s!

Volunteer computing

Idea: ask folks on the internet to help you compute a tricky problem on their desktops

- Protein folding - https://foldingathome.org/
- Telescope data analysis - seti@home
- Model-based RL for chess – https://lczero.org
  Fully asynchronous, volunteers donate compute to play games & submit results

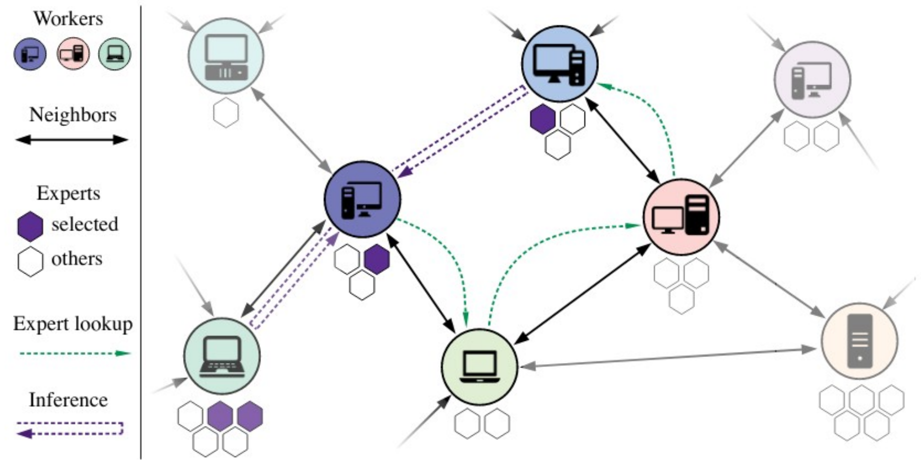
- General deep learning – limited success
   Kijsipongse E. et al, 2018, JSC
- Deep learning over torrents – yarr!
  https://arxiv.org/abs/2002.04013



Weird one: Learning@home

TL;DR

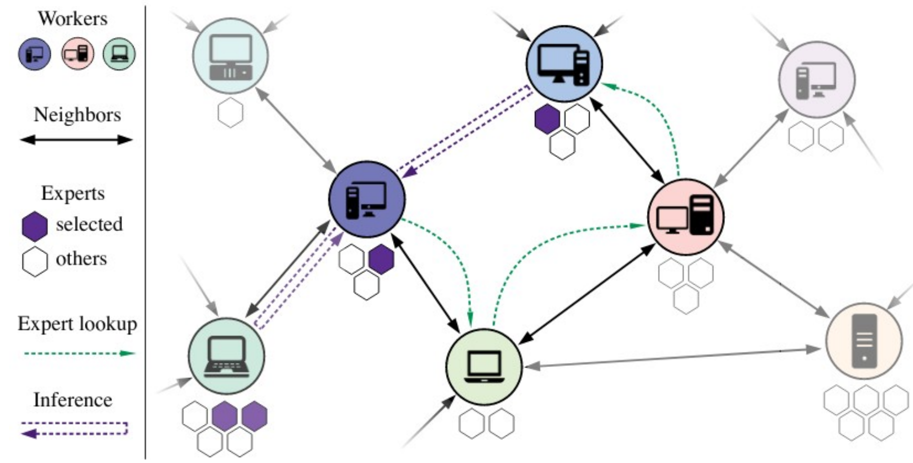
- each layer is a mixture of experts
- 1000s of distributed experts
  - only top-k experts are chosen depending on input features
- Fully asynchronous training
- Fault tolerance = dropout

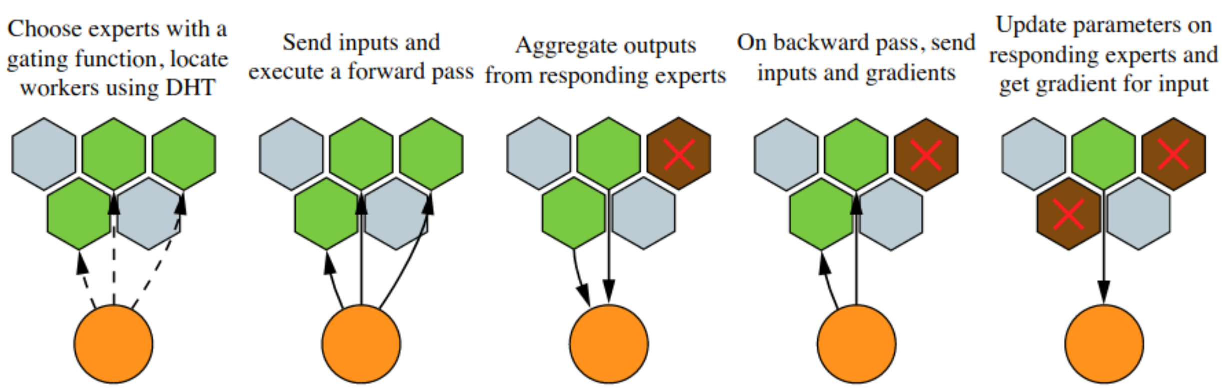


Weird one: Learning@home

TL;DR

- each layer is a mixture of experts
- 1000s of distributed experts
  - only top-k experts are chosen depending on input features
- Fully asynchronous training
- Fault tolerance = dropout





Read more:
arxiv.org/abs/2002.04013

Coolest part:
They use torrents for communication!