

Data-Parallel Deep Learning

Efficient DL, Episode III, 2022

Yandex
Research

LAMBDA 



Large problems need large models

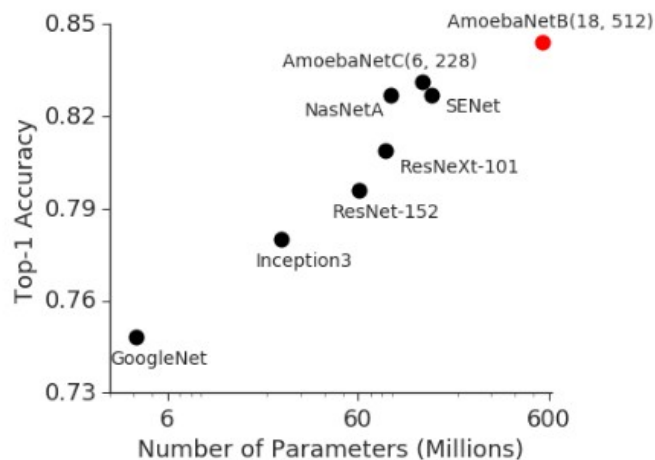
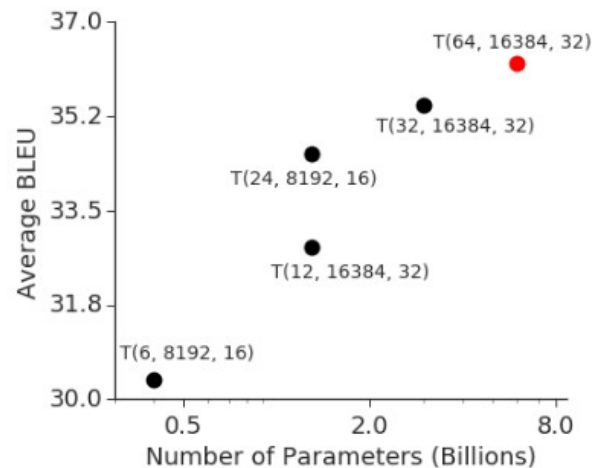


Image Classification
ImageNet

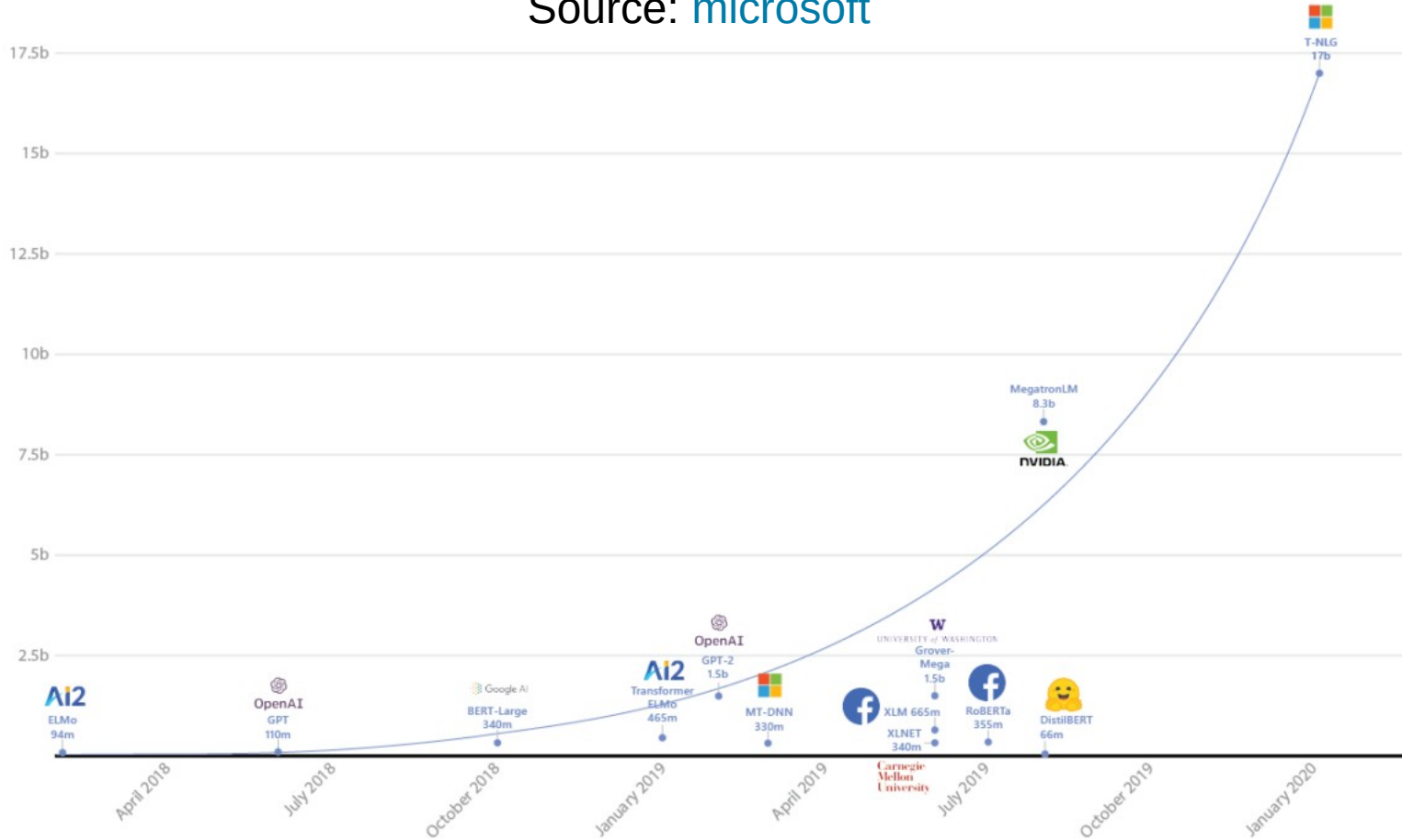


Machine Translation
average over WMT

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

Host

CPU

model



Devices

.cuda()

GPU1



Data-parallel training (naive)

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

Host

CPU

model

θ

batch

x

...

Devices

.cuda()

.cuda()

GPU1

θ

x

\Rightarrow

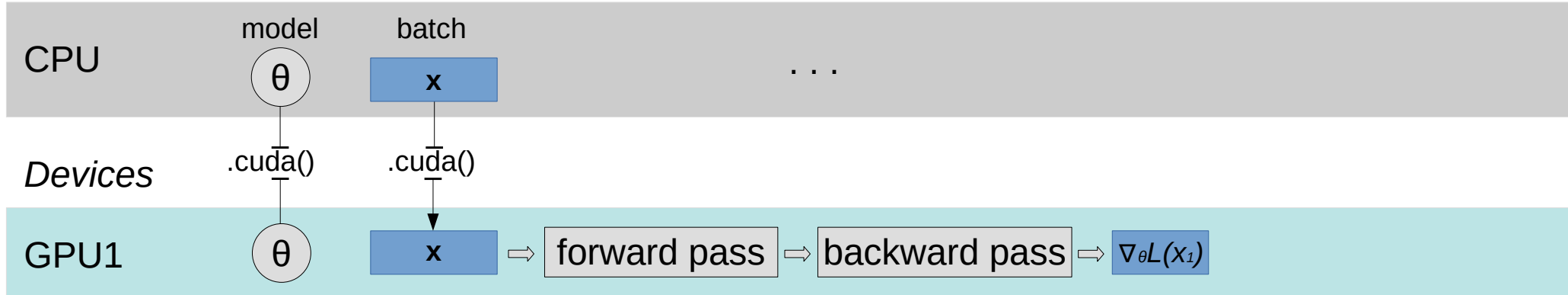
forward pass

\Rightarrow

backward pass

\Rightarrow

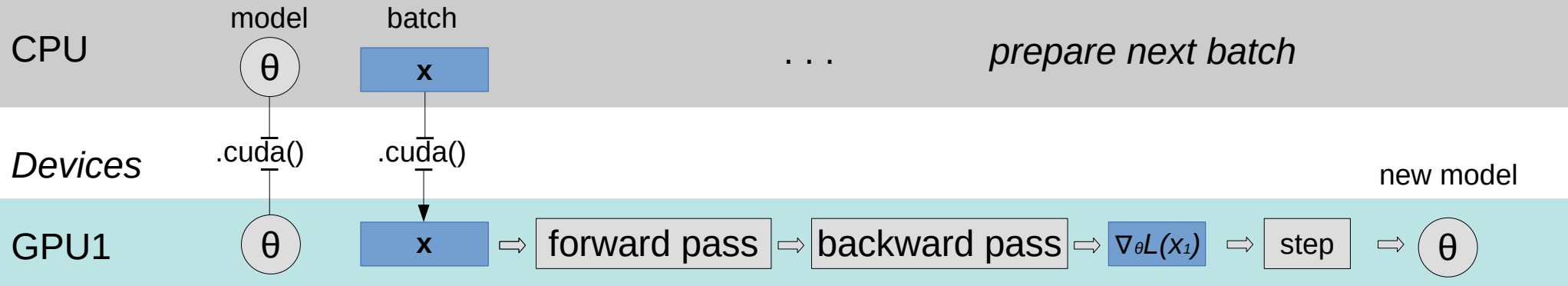
$\nabla_{\theta} L(x_1)$



Data-parallel training (naive)

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

Host



Data-parallel training (naive)

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

Host

CPU

model

θ

Devices

replicate

GPU1

θ

GPU2

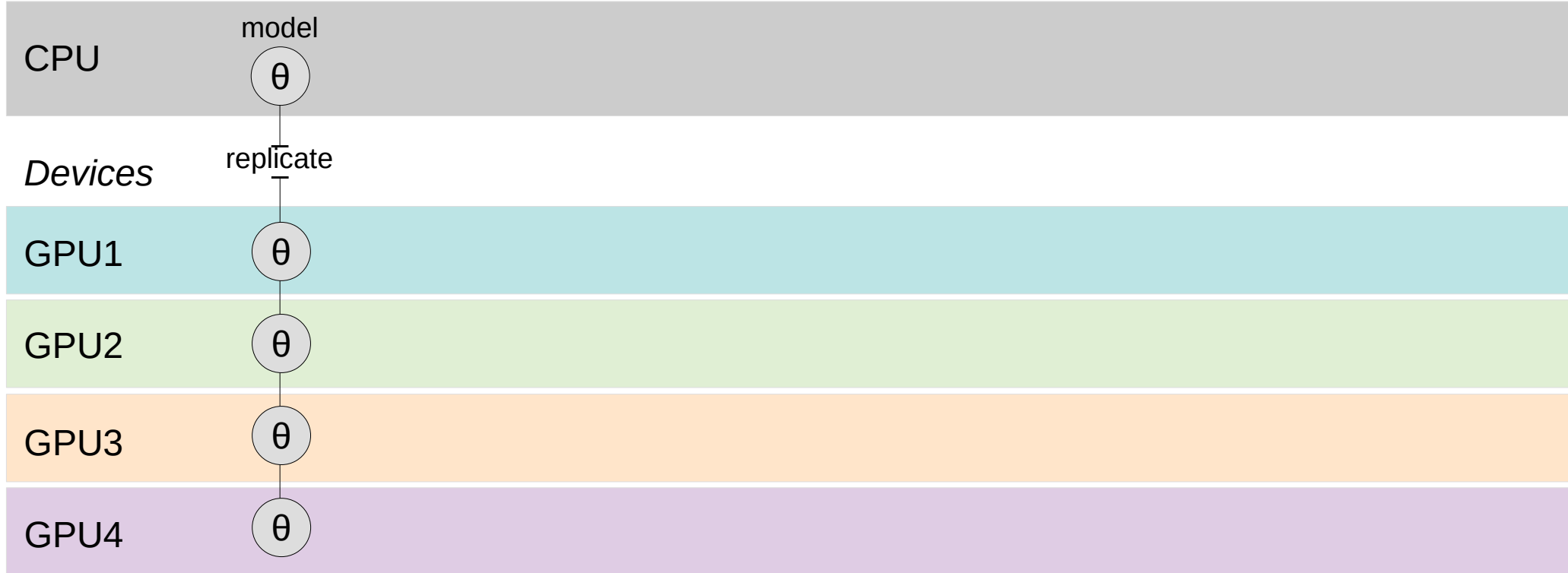
θ

GPU3

θ

GPU4

θ



Data-parallel training (naive)

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

Host

CPU

model

θ

batch

x_1 x_2 x_3 x_4

Devices

replicate

scatter

GPU1

θ

x_1

GPU2

θ

x_2

GPU3

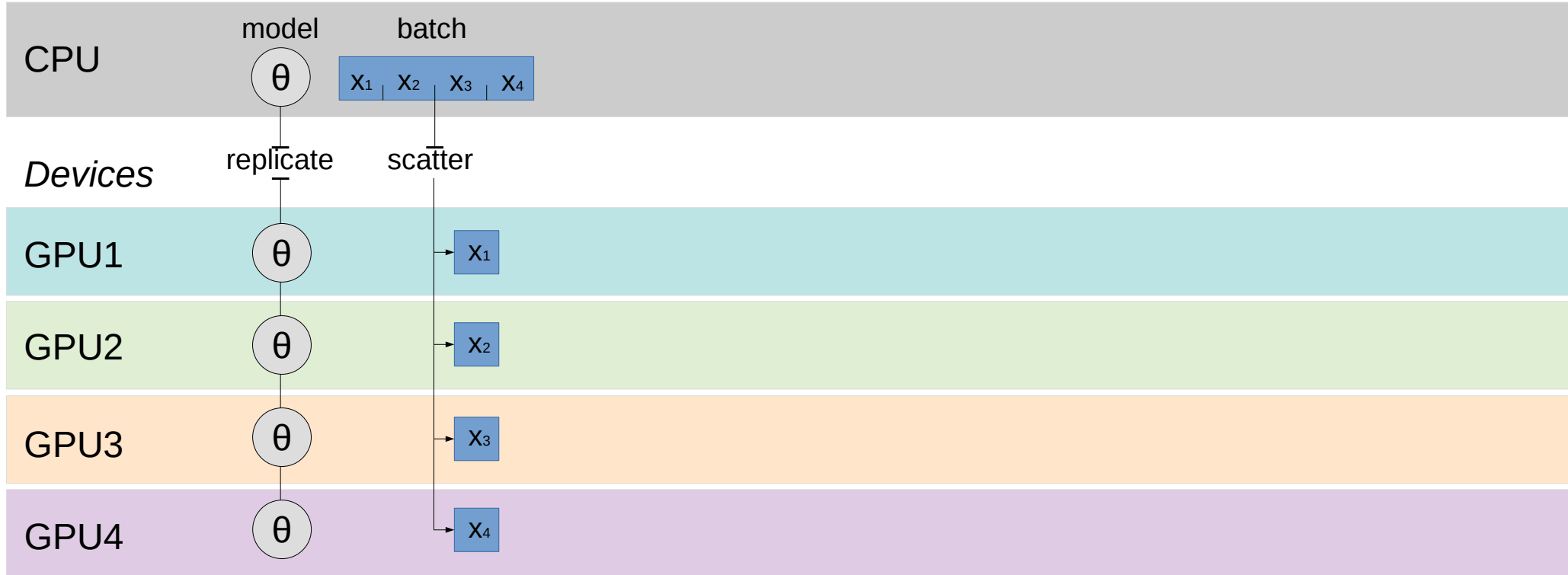
θ

x_3

GPU4

θ

x_4



Data-parallel training (naive)

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

Host

CPU

model

θ

batch

x_1 x_2 x_3 x_4

...

Devices

replicate

scatter

GPU1

θ

$x_1 \Rightarrow$ forward pass \Rightarrow backward pass $\Rightarrow \nabla_{\theta} L(x_1)$

GPU2

θ

$x_2 \Rightarrow$ forward pass \Rightarrow backward pass $\Rightarrow \nabla_{\theta} L(x_2)$

GPU3

θ

$x_3 \Rightarrow$ forward pass \Rightarrow backward pass $\Rightarrow \nabla_{\theta} L(x_3)$

GPU4

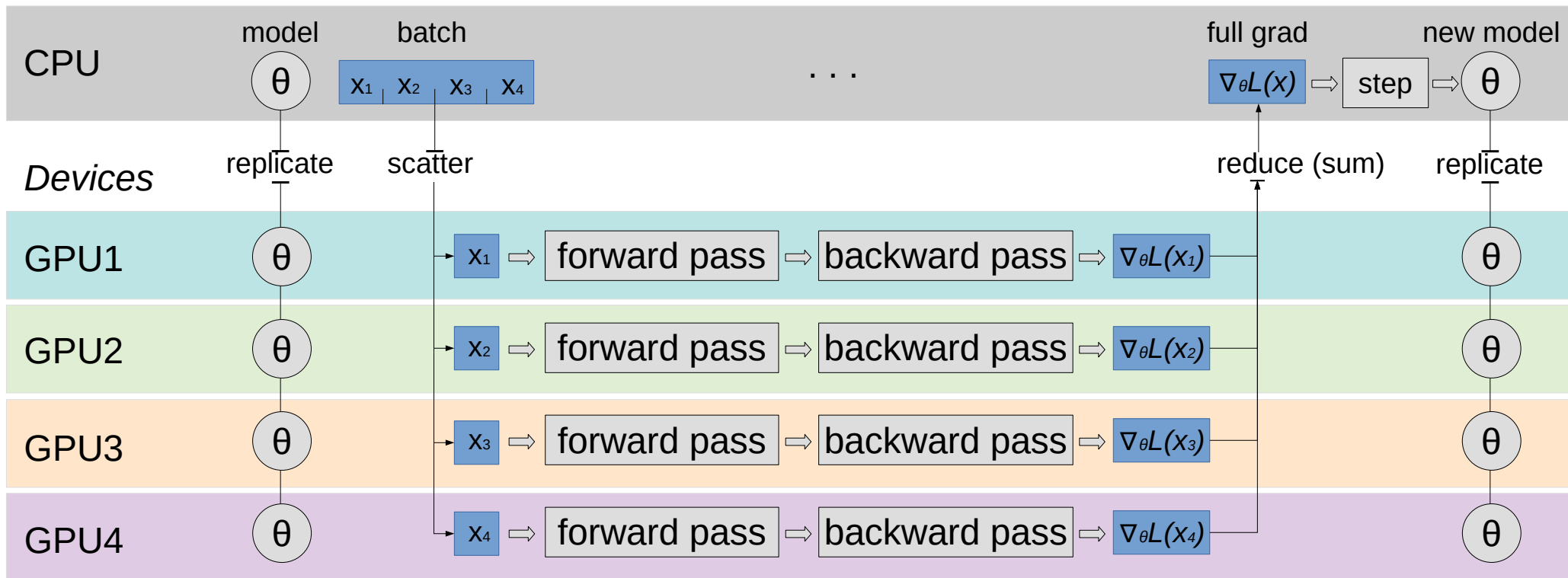
θ

$x_4 \Rightarrow$ forward pass \Rightarrow backward pass $\Rightarrow \nabla_{\theta} L(x_4)$

Data-parallel training (naive)

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

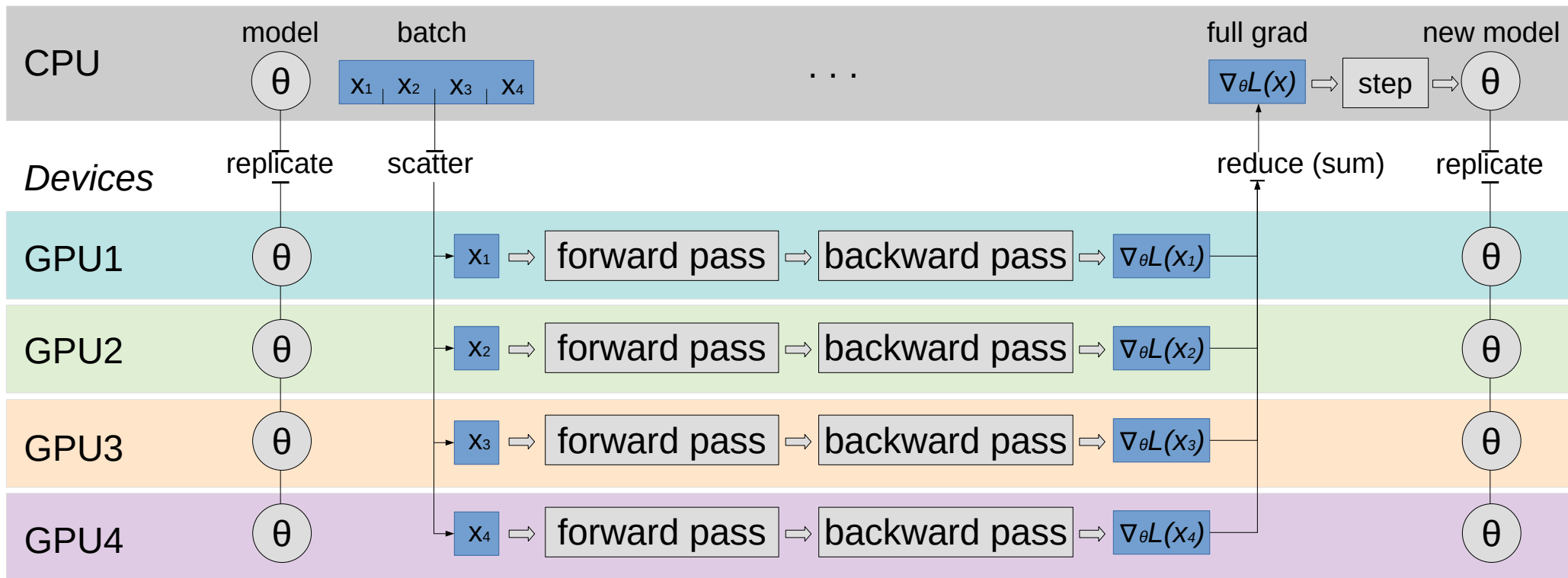
Host



Data-parallel training (naive)

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

Host

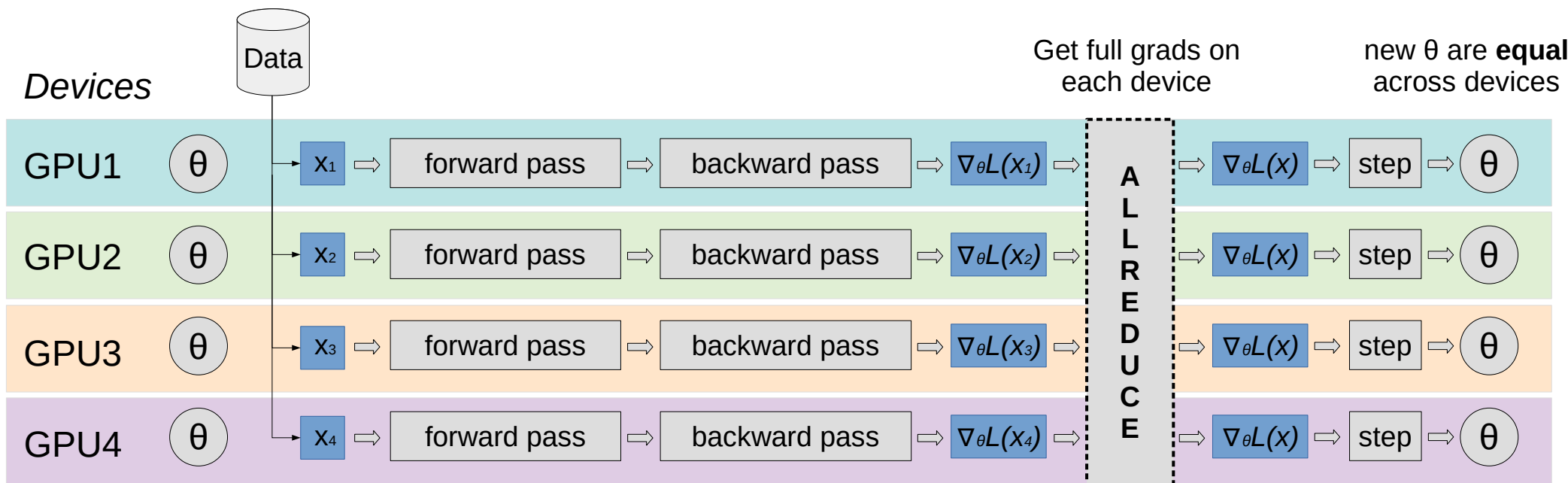


All-Reduce data parallel

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?

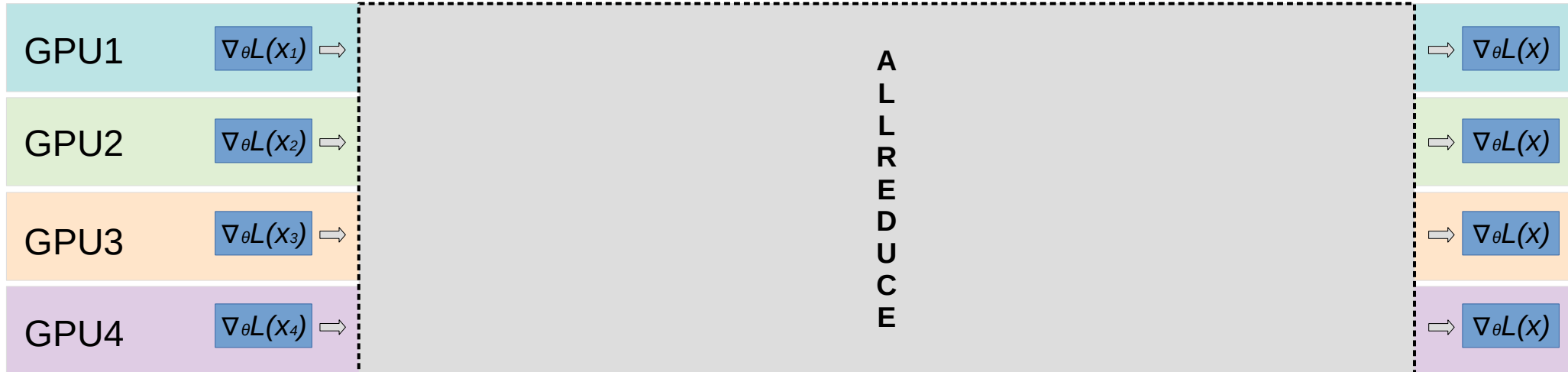


Faster allreduce

Input: each device has its own vector

Output: each device gets a sum of all vectors

Devices



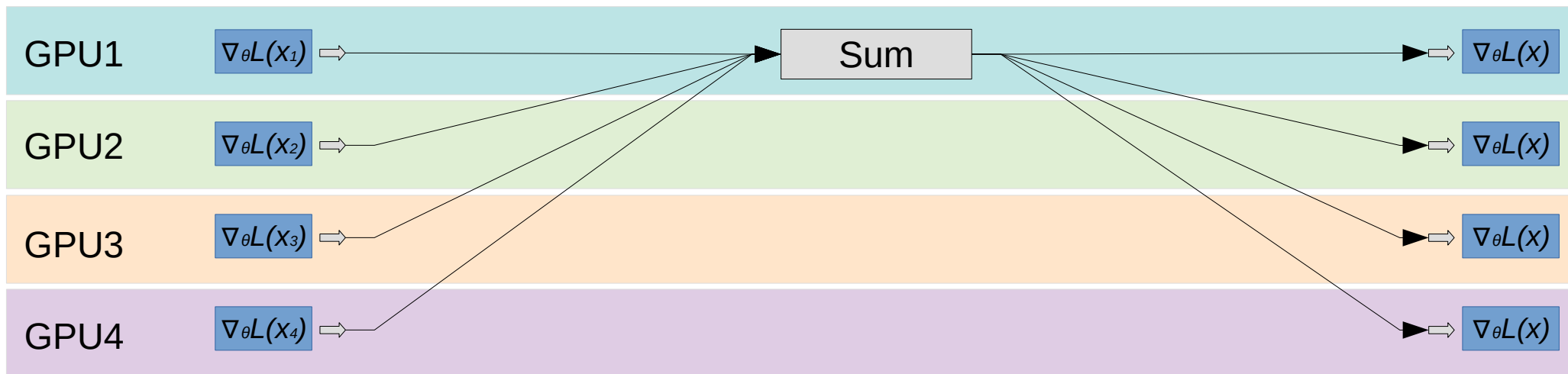
Faster allreduce

Input: each device has its own vector

Output: each device gets a sum of all vectors

Naive implementation

Devices



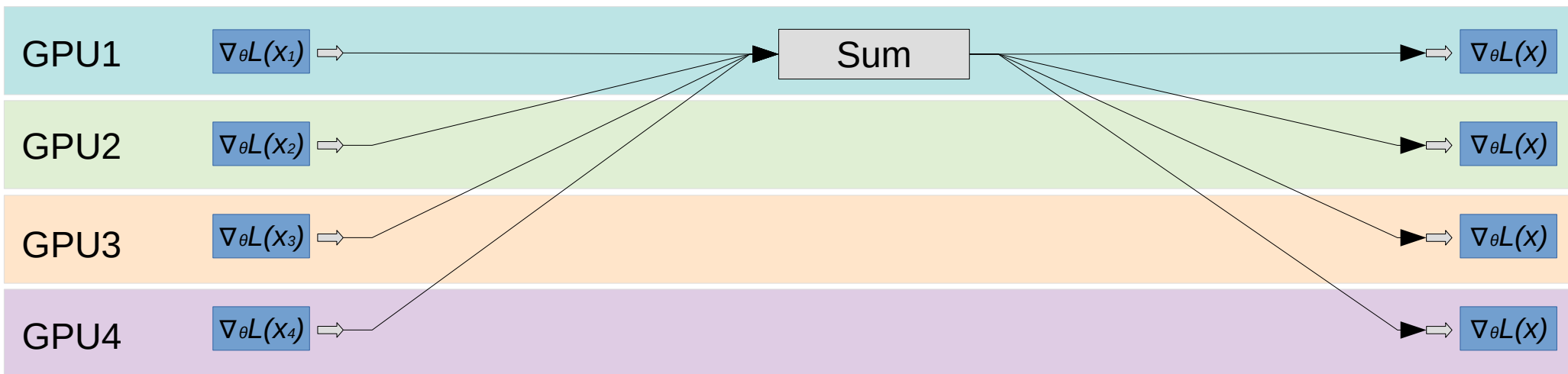
Faster allreduce

Input: each device has its own vector

Output: each device gets a sum of all vectors

Q: Can we do better?

Devices



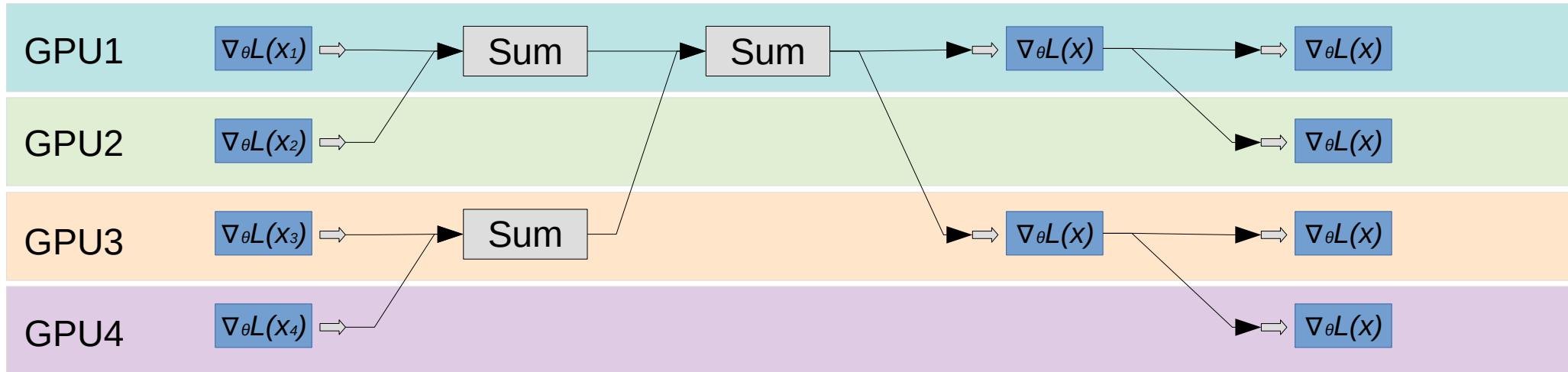
Faster allreduce

Input: each device has its own vector

Output: each device gets a sum of all vectors

Tree-allreduce

Devices



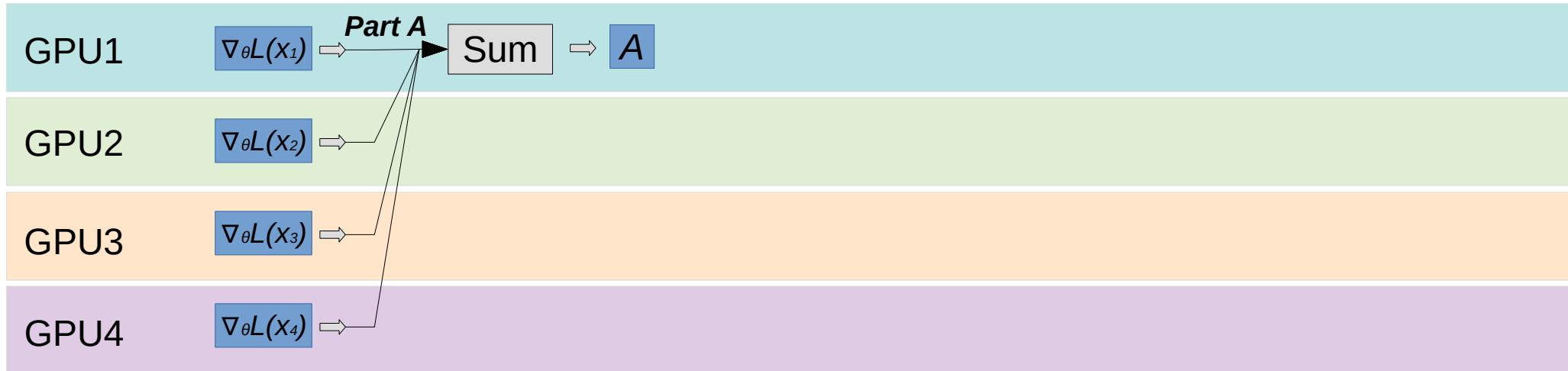
Faster allreduce

Input: each device has its own vector

Output: each device gets a sum of all vectors

Butterfly-allreduce – split data into chunks (ABCD)

Devices



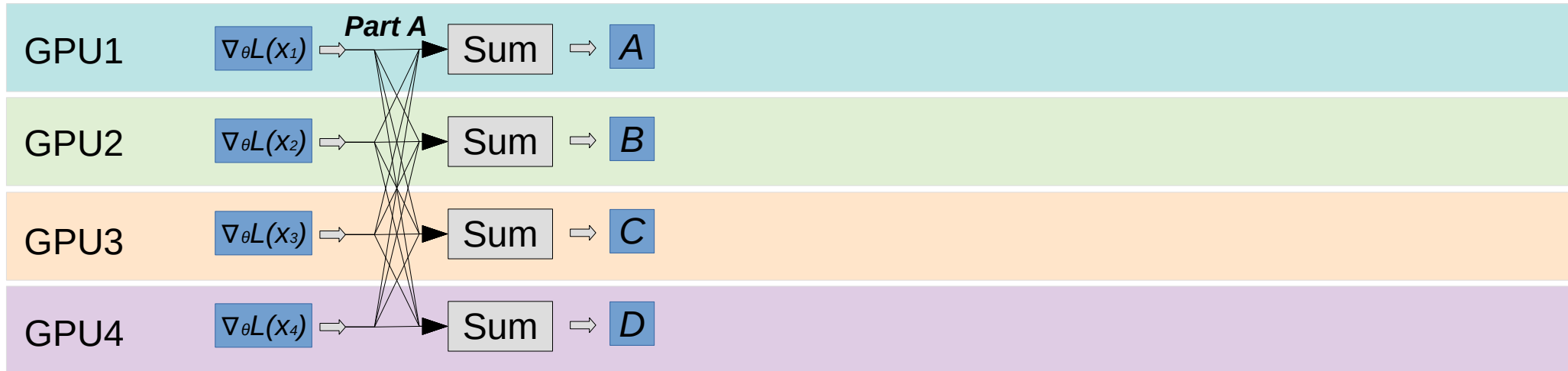
Faster allreduce

Input: each device has its own vector

Output: each device gets a sum of all vectors

Butterfly-allreduce – split data into chunks (ABCD)

Devices



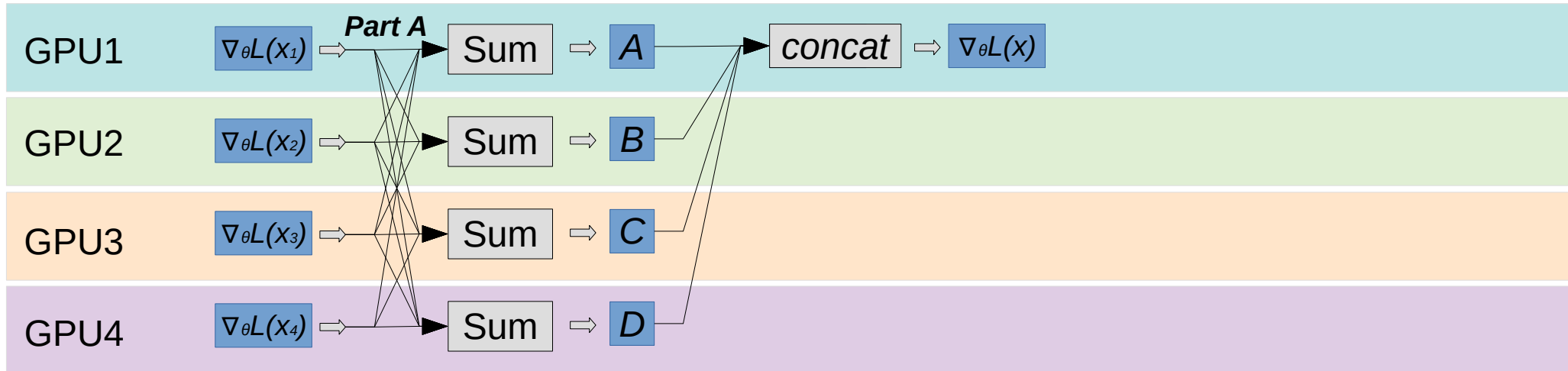
Faster allreduce

Input: each device has its own vector

Output: each device gets a sum of all vectors

Butterfly-allreduce – split data into chunks (ABCD)

Devices



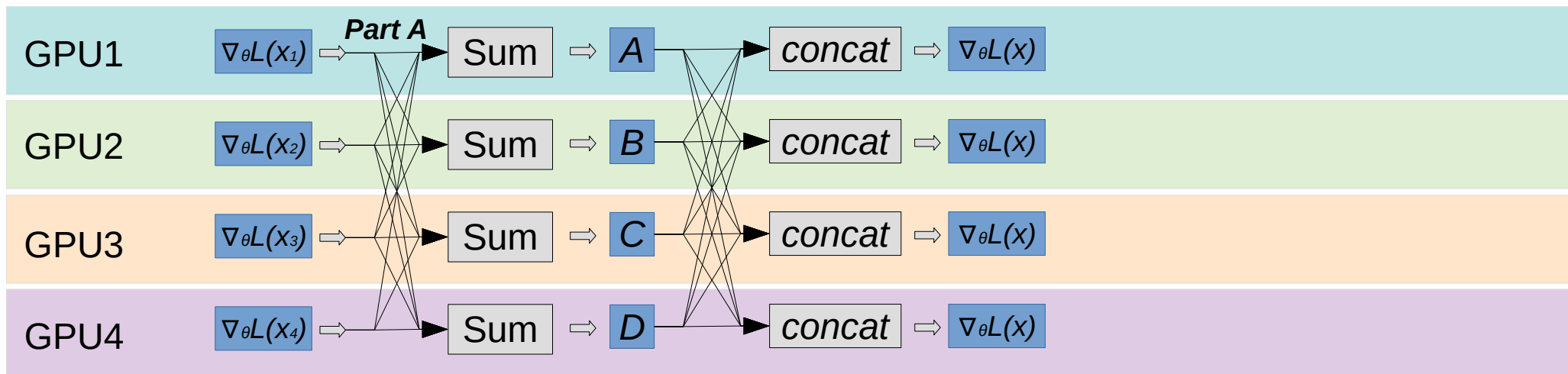
Faster allreduce

Input: each device has its own vector

Output: each device gets a sum of all vectors

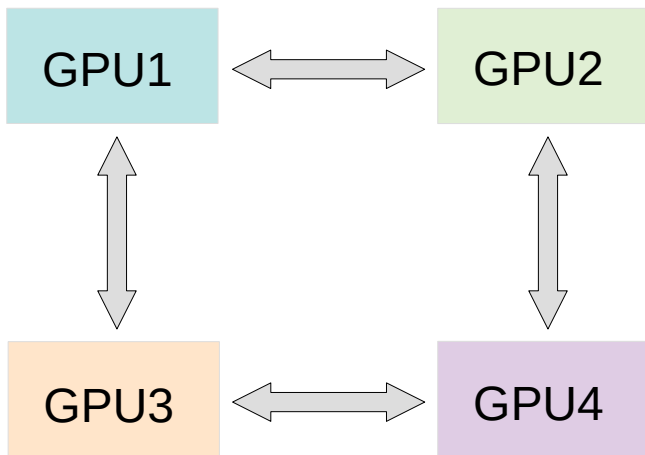
Ring-allreduce – split data into chunks (ABCD)

Devices



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

Ring allreduce

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

[Time to use the whiteboard]

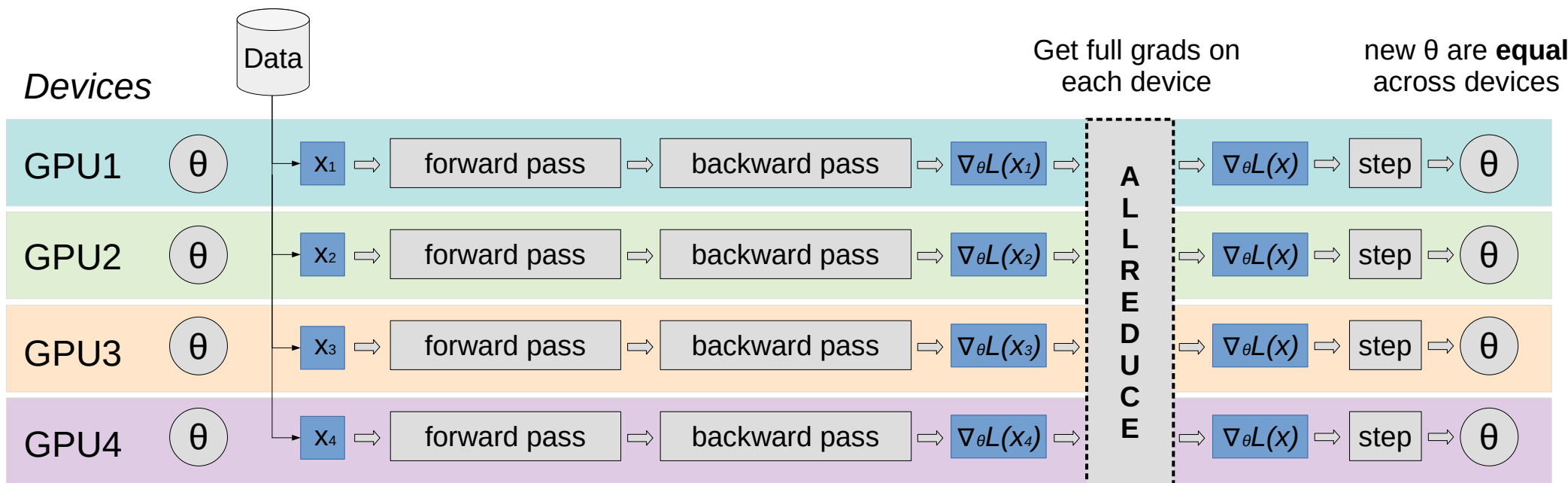
Answer & more: tinyurl.com/ring-allreduce-blog

All-Reduce 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?

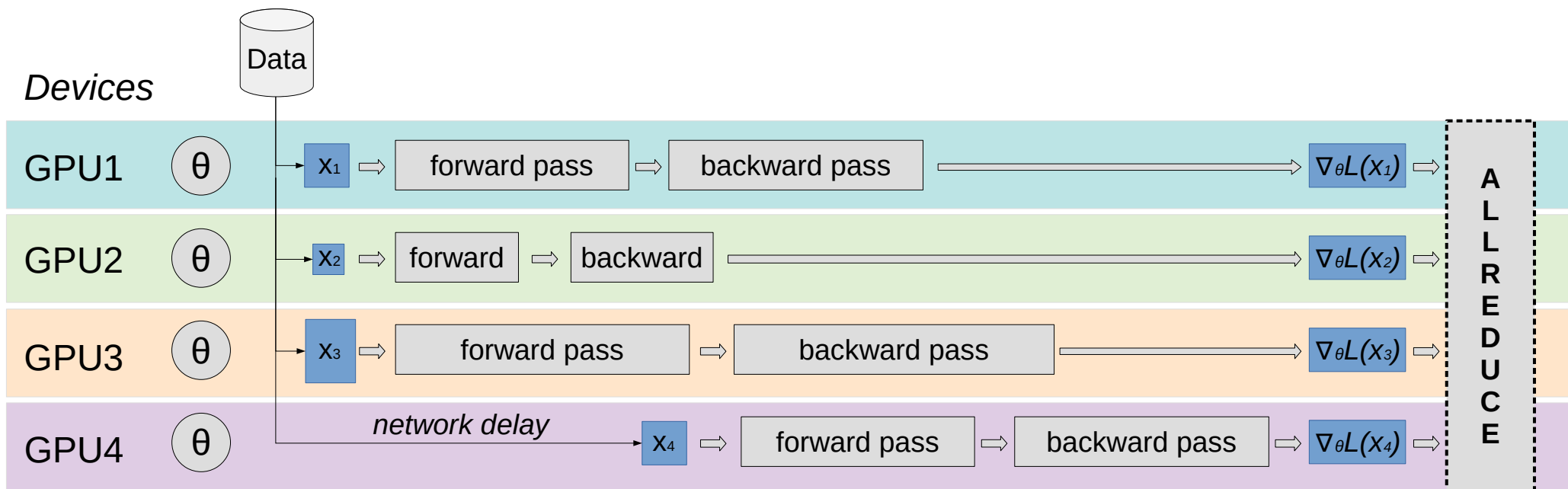


All-Reduce data parallel VS reality

arxiv.org/abs/1706.02677

Each gpu has different processing time & delays

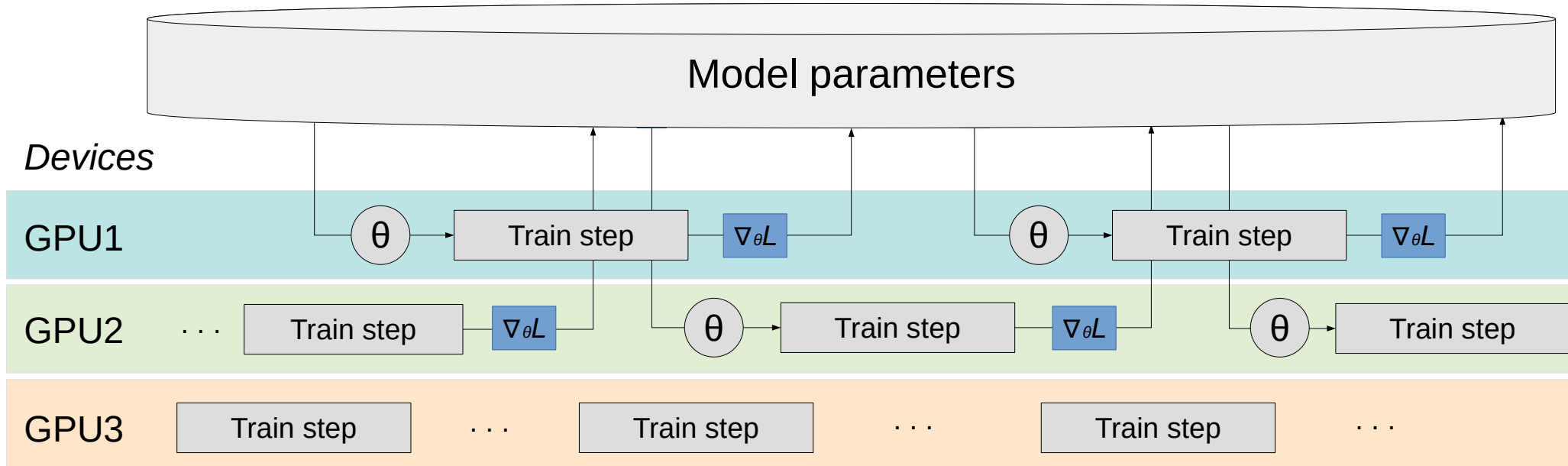
Q: can we improve device utilization?



Recap: Parameter Server

HOGWILD! arxiv.org/abs/1106.5730

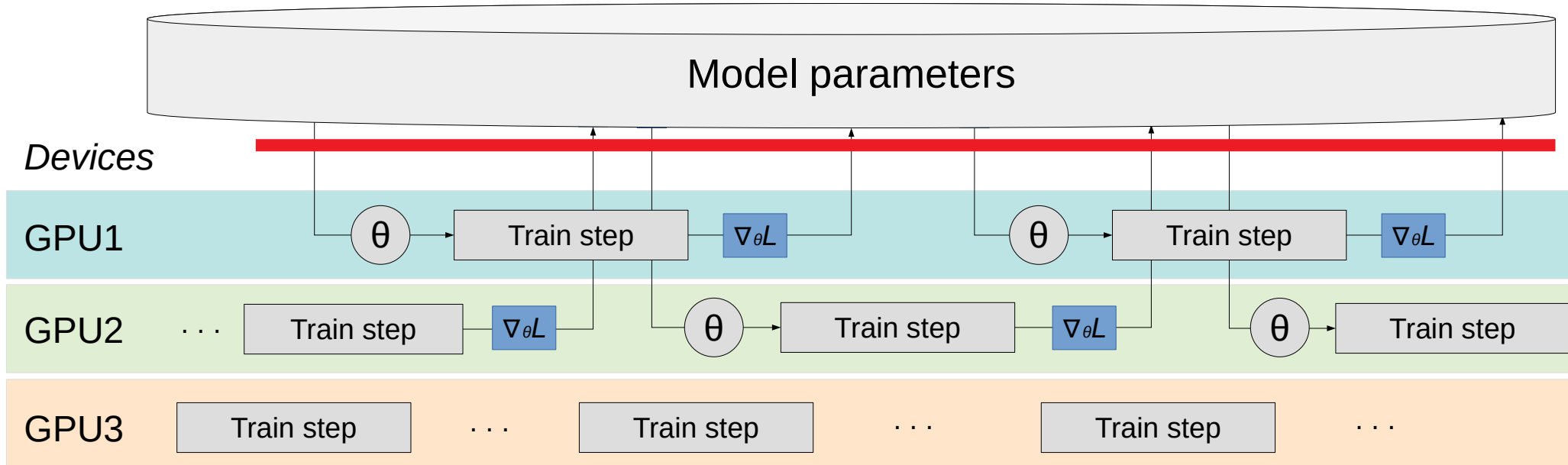
Idea: remove synchronization step altogether, use parameter server



Recap: Parameter Server

HOGWILD! arxiv.org/abs/1106.5730

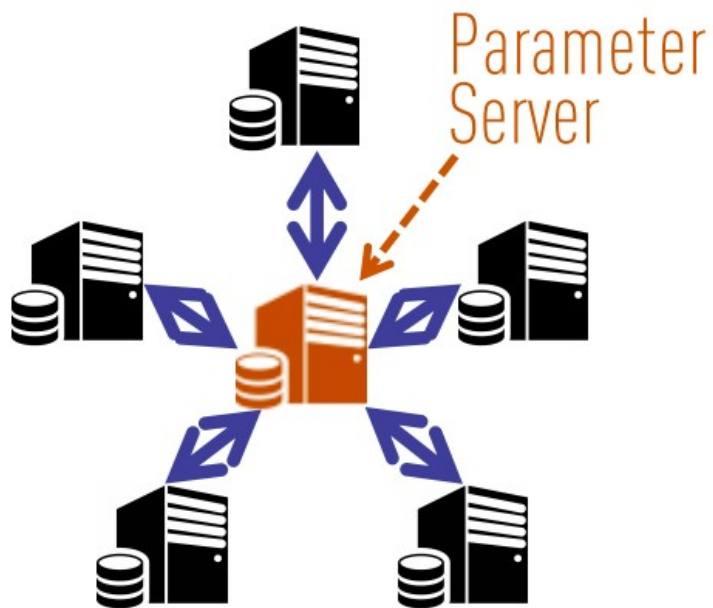
Idea: remove synchronization step altogether, use parameter server



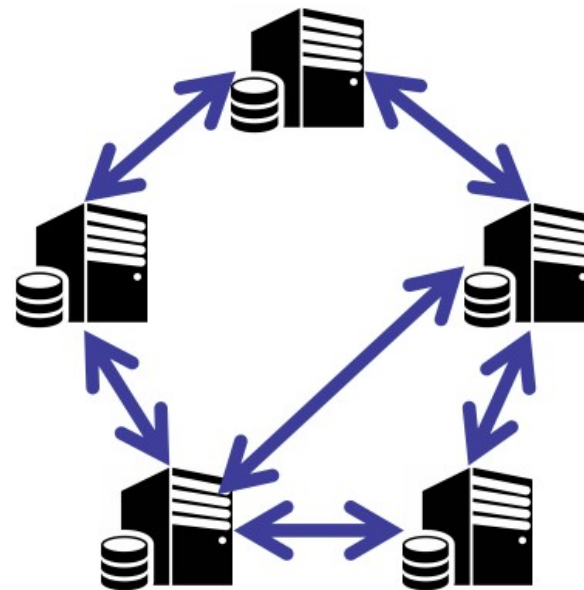
Problem: parameter servers need to ingest tons of data over training

Decentralized Training with Gossip

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



(a) Centralized Topology



(b) Decentralized Topology

Decentralized Training with Gossip

Source: <https://tinyurl.com/can-dsgd-outperform>

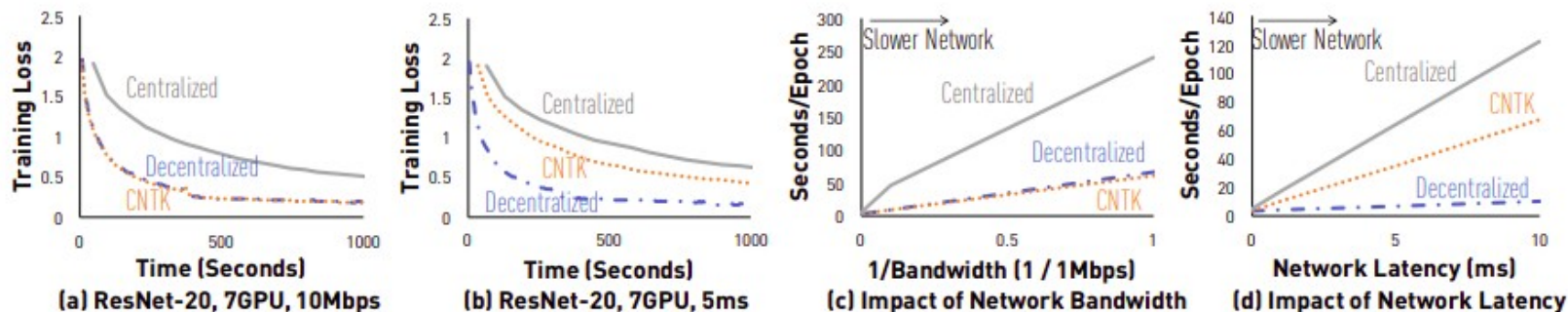


Figure 2: Comparison between D-PSGD and two centralized implementations (7 and 10 GPUs).

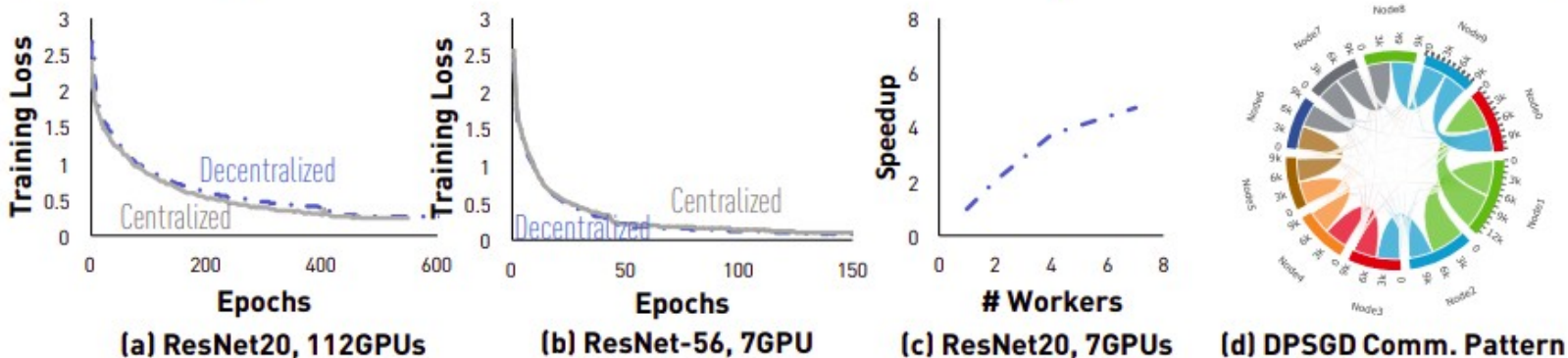
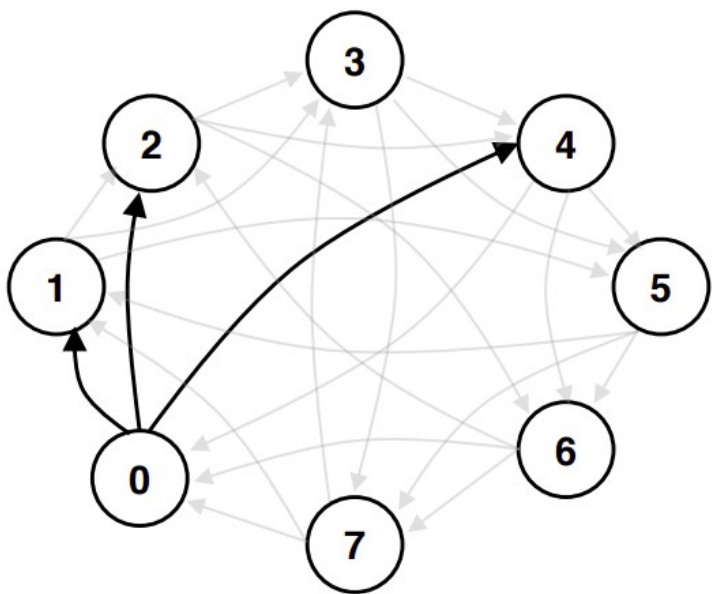


Figure 3: (a) Convergence Rate; (b) D-PSGD Speedup; (c) D-PSGD Communication Patterns.

Stochastic Gradient Push

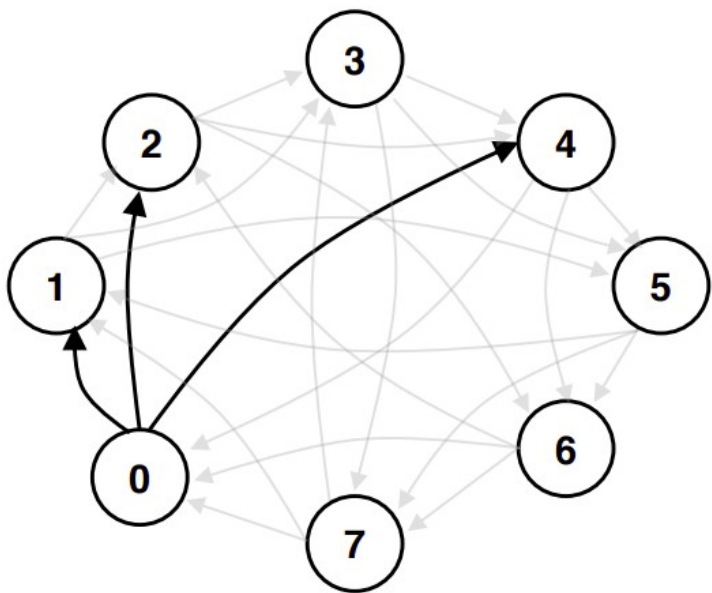
Source: <https://arxiv.org/abs/1811.10792>



(a) Directed Exponential Graph highlighting node 0's out-neighbours

Stochastic Gradient Push

Source: <https://arxiv.org/abs/1811.10792>



(a) Directed Exponential Graph highlighting node 0's out-neighbours

Algorithm 1 Stochastic Gradient Push (SGP)

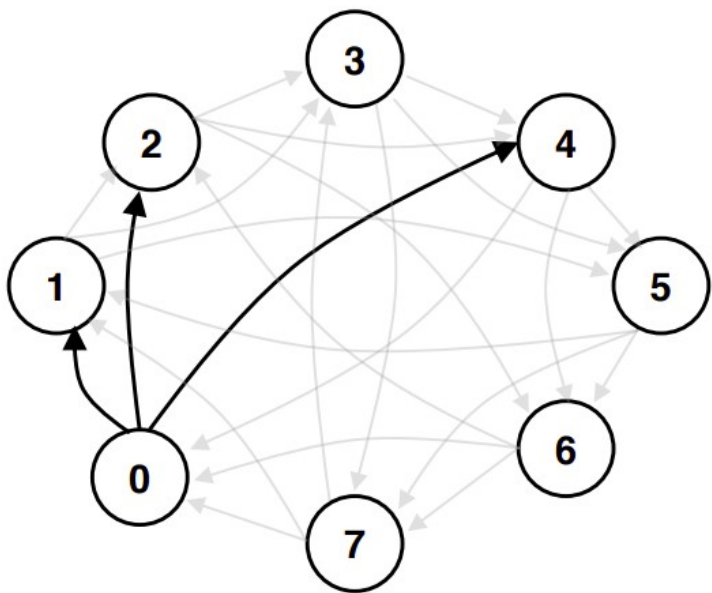
Require: Initialize $\gamma > 0$, $\mathbf{x}_i^{(0)} = \mathbf{z}_i^{(0)} \in \mathbb{R}^d$ and $w_i^{(0)} = 1$ for all nodes $i \in \{1, 2, \dots, n\}$

- 1: **for** $k = 0, 1, 2, \dots, K$, at node i , **do**
- 2: Sample new mini-batch $\xi_i^{(k)} \sim \mathcal{D}_i$ from local distribution
- 3: Compute mini-batch gradient at $\mathbf{z}_i^{(k)}$: $\nabla F_i(\mathbf{z}_i^{(k)}; \xi_i^{(k)})$

<to be continued>

Stochastic Gradient Push

Source: <https://arxiv.org/abs/1811.10792>



(a) Directed Exponential Graph highlighting node 0's out-neighbours

Algorithm 1 Stochastic Gradient Push (SGP)

Require: Initialize $\gamma > 0$, $\mathbf{x}_i^{(0)} = \mathbf{z}_i^{(0)} \in \mathbb{R}^d$ and $w_i^{(0)} = 1$ for all nodes $i \in \{1, 2, \dots, n\}$

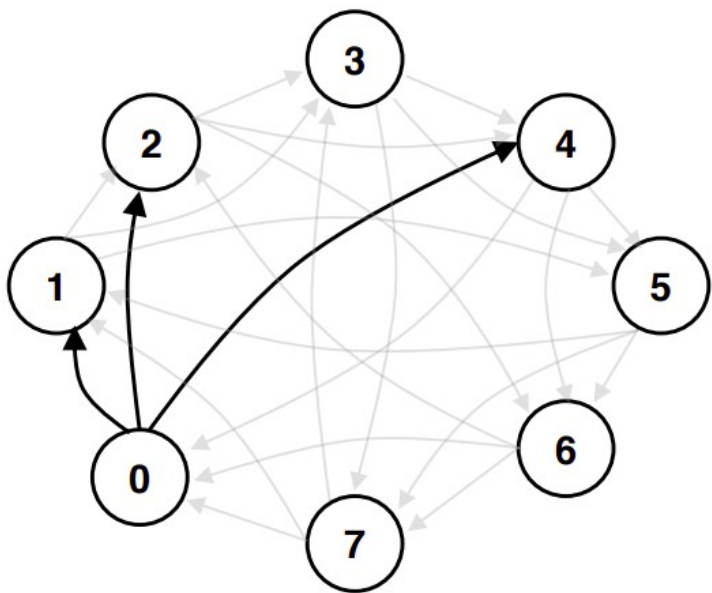
- 1: **for** $k = 0, 1, 2, \dots, K$, at node i , **do**
- 2: Sample new mini-batch $\xi_i^{(k)} \sim \mathcal{D}_i$ from local distribution
- 3: Compute mini-batch gradient at $\mathbf{z}_i^{(k)}$: $\nabla F_i(\mathbf{z}_i^{(k)}; \xi_i^{(k)})$
- 4: $\mathbf{x}_i^{(k+\frac{1}{2})} = \mathbf{x}_i^{(k)} - \gamma \nabla F_i(\mathbf{z}_i^{(k)}; \xi_i^{(k)})$

normal GD step

<to be continued>

Stochastic Gradient Push

Source: <https://arxiv.org/abs/1811.10792>



(a) Directed Exponential Graph highlighting node 0's out-neighbours

Algorithm 1 Stochastic Gradient Push (SGP)

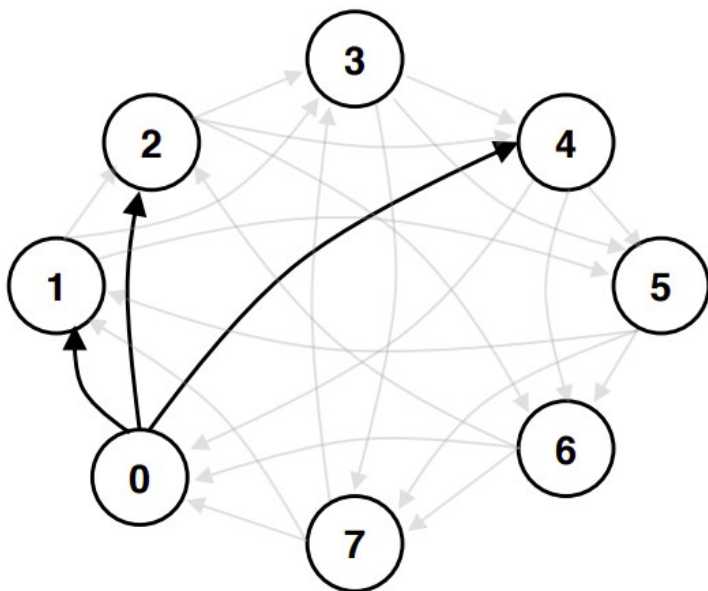
Require: Initialize $\gamma > 0$, $\mathbf{x}_i^{(0)} = \mathbf{z}_i^{(0)} \in \mathbb{R}^d$ and $w_i^{(0)} = 1$ for all nodes $i \in \{1, 2, \dots, n\}$

- 1: **for** $k = 0, 1, 2, \dots, K$, at node i , **do**
- 2: Sample new mini-batch $\xi_i^{(k)} \sim \mathcal{D}_i$ from local distribution
- 3: Compute mini-batch gradient at $\mathbf{z}_i^{(k)}$: $\nabla \mathbf{F}_i(\mathbf{z}_i^{(k)}; \xi_i^{(k)})$
- 4: $\mathbf{x}_i^{(k+\frac{1}{2})} = \mathbf{x}_i^{(k)} - \gamma \nabla \mathbf{F}_i(\mathbf{z}_i^{(k)}; \xi_i^{(k)})$
- 5: Send $(p_{j,i}^{(k)} \mathbf{x}_i^{(k+\frac{1}{2})}, p_{j,i}^{(k)} w_i^{(k)})$ to out-neighbors;
 receive $(p_{i,j}^{(k)} \mathbf{x}_j^{(k+\frac{1}{2})}, p_{i,j}^{(k)} w_j^{(k)})$ from in-neighbors

<to be continued>

Stochastic Gradient Push

Source: <https://arxiv.org/abs/1811.10792>



(a) Directed Exponential Graph highlighting node 0's out-neighbours

Algorithm 1 Stochastic Gradient Push (SGP)

Require: Initialize $\gamma > 0$, $\mathbf{x}_i^{(0)} = \mathbf{z}_i^{(0)} \in \mathbb{R}^d$ and $w_i^{(0)} = 1$ for all nodes $i \in \{1, 2, \dots, n\}$

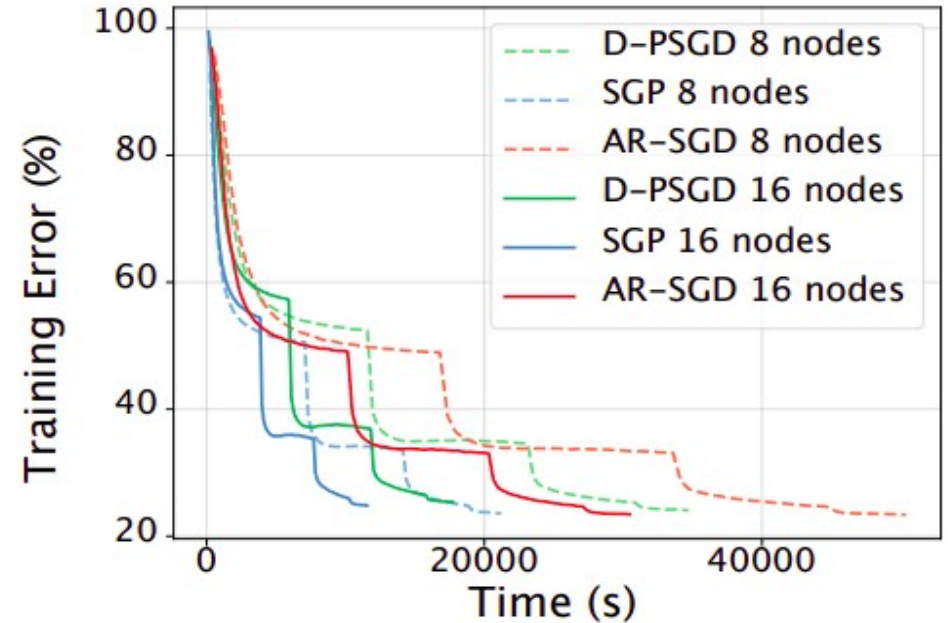
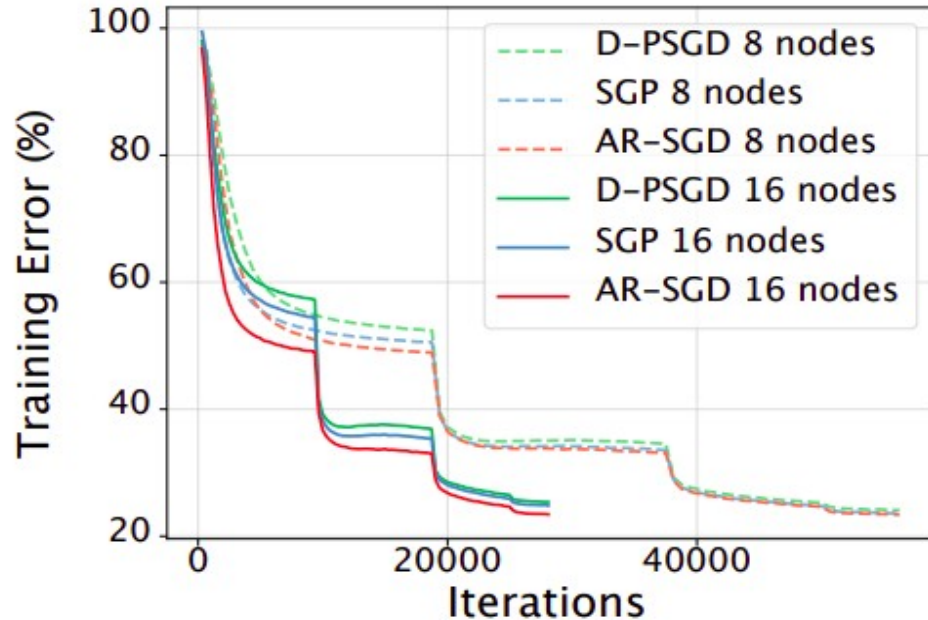
- 1: **for** $k = 0, 1, 2, \dots, K$, at node i , **do**
- 2: Sample new mini-batch $\xi_i^{(k)} \sim \mathcal{D}_i$ from local distribution
- 3: Compute mini-batch gradient at $\mathbf{z}_i^{(k)}$: $\nabla \mathbf{F}_i(\mathbf{z}_i^{(k)}; \xi_i^{(k)})$
- 4: $\mathbf{x}_i^{(k+\frac{1}{2})} = \mathbf{x}_i^{(k)} - \gamma \nabla \mathbf{F}_i(\mathbf{z}_i^{(k)}; \xi_i^{(k)})$
- 5: Send $(p_{j,i}^{(k)} \mathbf{x}_i^{(k+\frac{1}{2})}, p_{j,i}^{(k)} w_i^{(k)})$ to out-neighbors;
 receive $(p_{i,j}^{(k)} \mathbf{x}_j^{(k+\frac{1}{2})}, p_{i,j}^{(k)} w_j^{(k)})$ from in-neighbors
- 6: $\mathbf{x}_i^{(k+1)} = \sum_j p_{i,j}^{(k)} \mathbf{x}_j^{(k+\frac{1}{2})}$
- 7: $w_i^{(k+1)} = \sum_j p_{i,j}^{(k)} w_j^{(k)}$
- 8: $\mathbf{z}_i^{(k+1)} = \mathbf{x}_i^{(k+1)} / w_i^{(k+1)}$
- 9: **end for**

**weighted
average**

Stochastic Gradient Push

Source: <https://arxiv.org/abs/1811.10792>

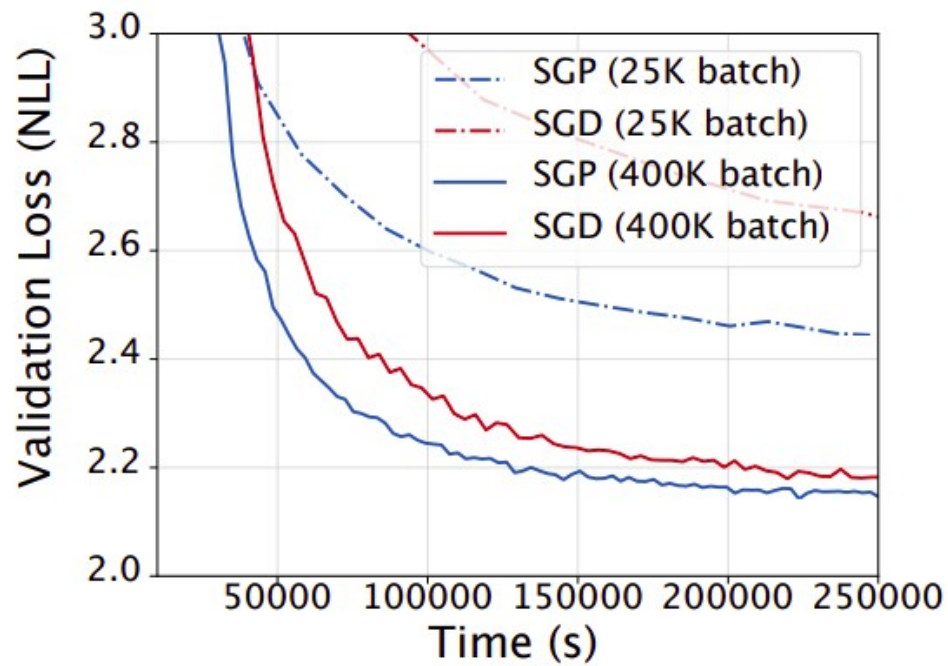
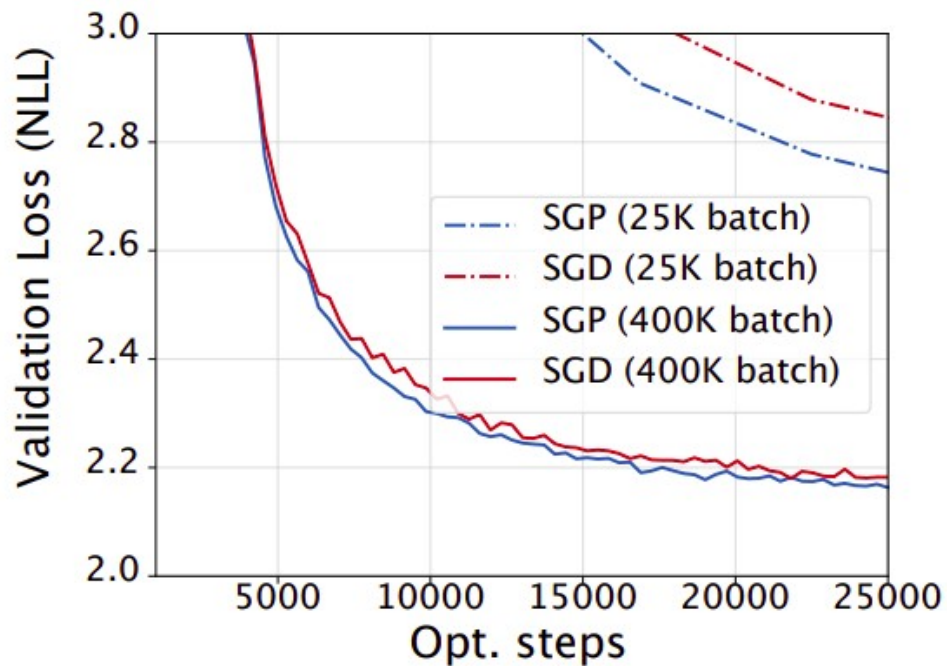
SGP vs ImageNet (ResNet50 + SGD w/ momentum)



Stochastic Gradient Push

Source: <https://arxiv.org/abs/1811.10792>

SGP vs WMT English-German (Transformer, Adam)



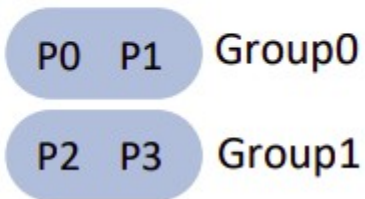
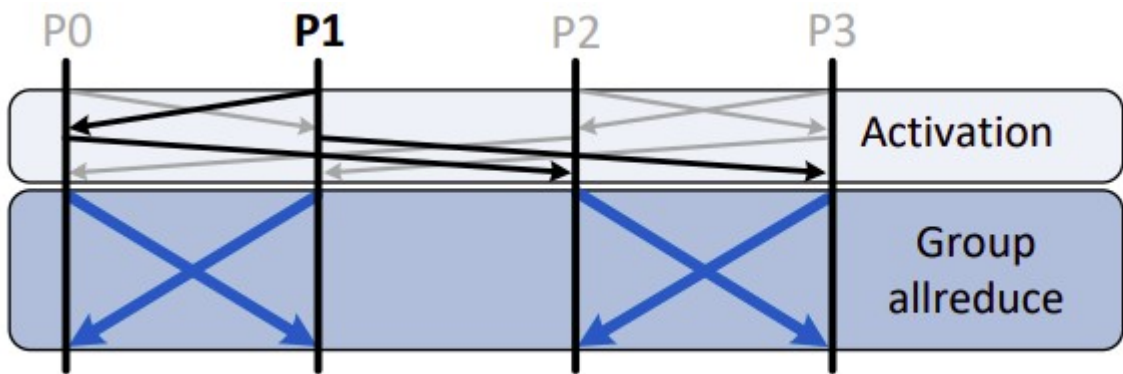
Gossip vs All-Reduce

Your thoughts?

Gossip + All-Reduce

Source: arxiv.org/abs/2005.00124

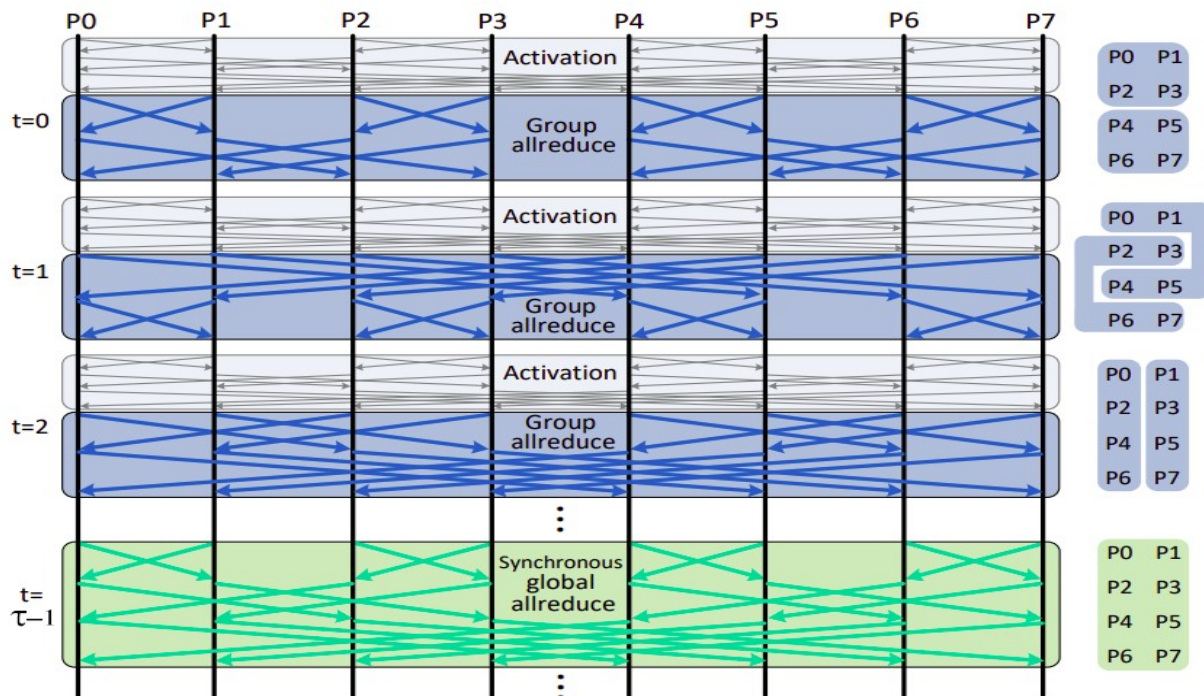
Core idea: run all-reduce in independent groups
You only have to synchronize for your small group
Swap groupmates between iterations



Gossip + All-Reduce

Source: arxiv.org/abs/2005.00124

Core idea: run all-reduce in independent groups
You only have to synchronize for your small group
Swap groupmates between iterations

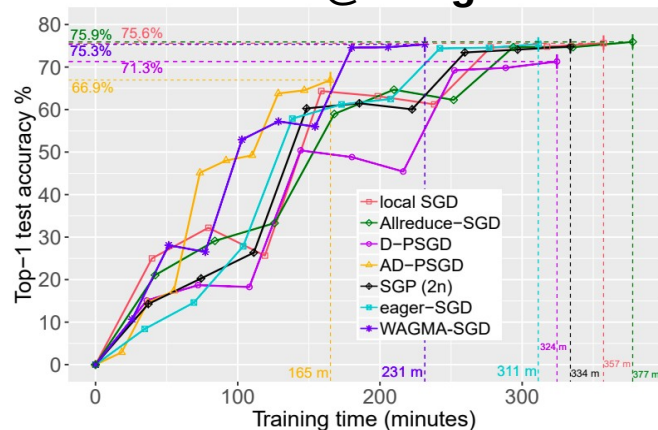


Gossip + All-Reduce

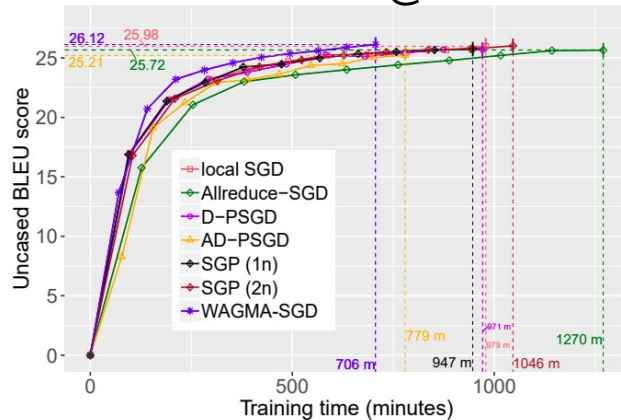
Source: arxiv.org/abs/2005.00124

Experiment setup: up to 1024 GPU,
Natural (or emulated) network latency

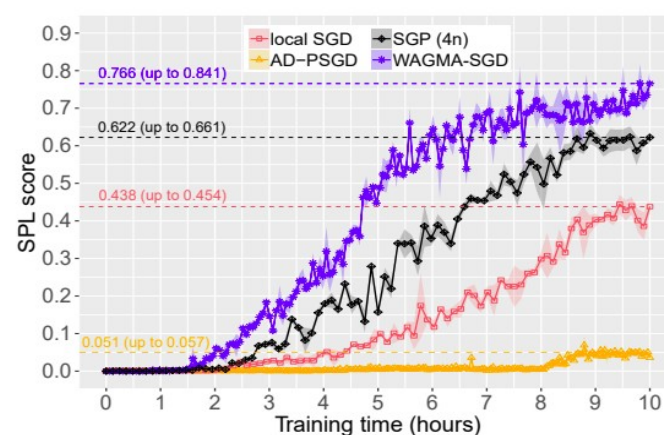
Image Classification
ResNet50 @ ImageNet



Machine Translation
Transformer @ WMT17



Reinforcement Learning
DDPO on Habitat



Side-quest: reducing network usage

Virtual batch / virtual pipeline

ёж, открой доску

</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>



"That's all Folks!"

Isberg®