Reducing Deep Learning training times with parallelism strategies and TensorFlow Distribute.

Juan Manuel Muñoz Betancur

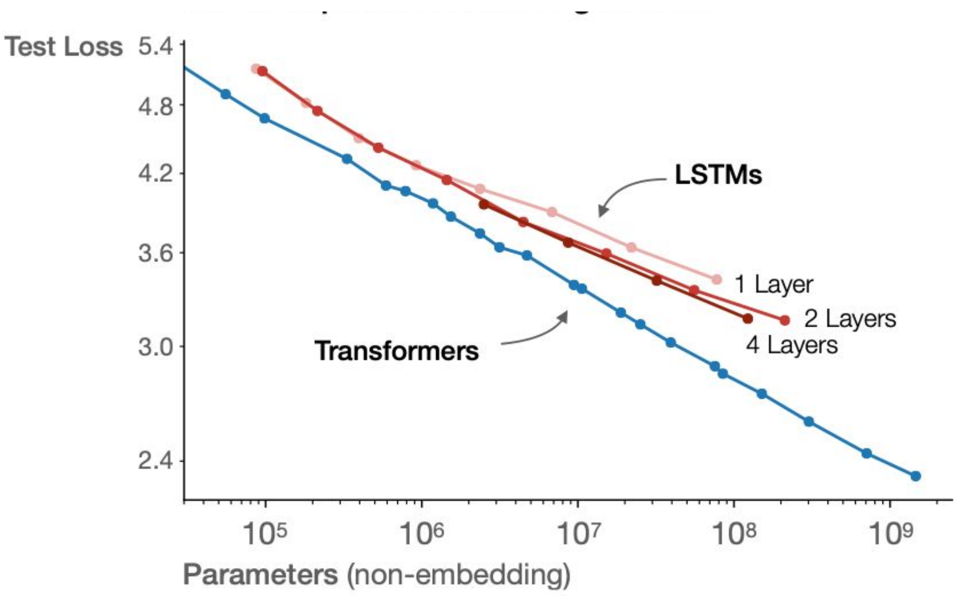
PRESENTATION **OUTLINE**

- Introduction/Motivation.
- Common types of parallelism.
  - Model Parallelism
  - Data Parallelism
- Tensorflow Demo.

Motivation

Get results faster and iterate quickly in your small/medium/large projects without additional costs.

State of the art models: GPT-3 has 175 Billion parameters, to train it even in the best SINGLE GPU available in the market would take about 355 years.



Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

Motivation

To train a model you need to take into account the size of the parameters, the forward pass activations (saving them for the backprop) and the gradients of the weights.

Params size
InceptionV3 24M params
@ fp64 = 24 x 10^6 x 8 bytes
= 192 MB

Activations size
InceptionV3 @ fp64 batch_size=256
~= 256 (4 x 10^6 x 8 bytes)
= 32MB * 256 **= 8GB**

Params size
GPT-3 175B parameters @fp16
= 175 x 10^9 x 2 bytes
= 316.2 GB

Activations size
GPT-3 **~TBs**

Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

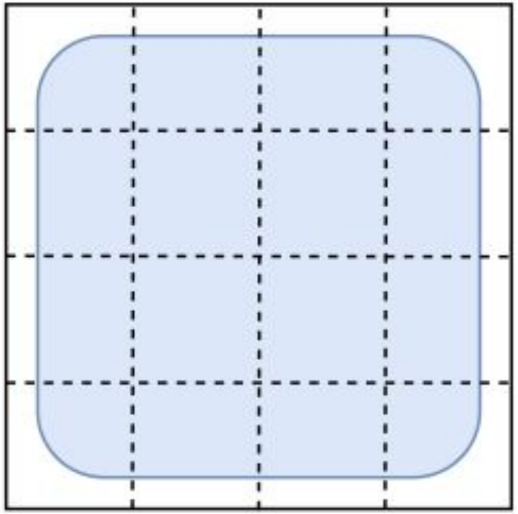
COMMON **TYPES OF** **PARALLELISM**

- Model Parallelism.
    - Pipeline Parallelism
- Data Parallelism.

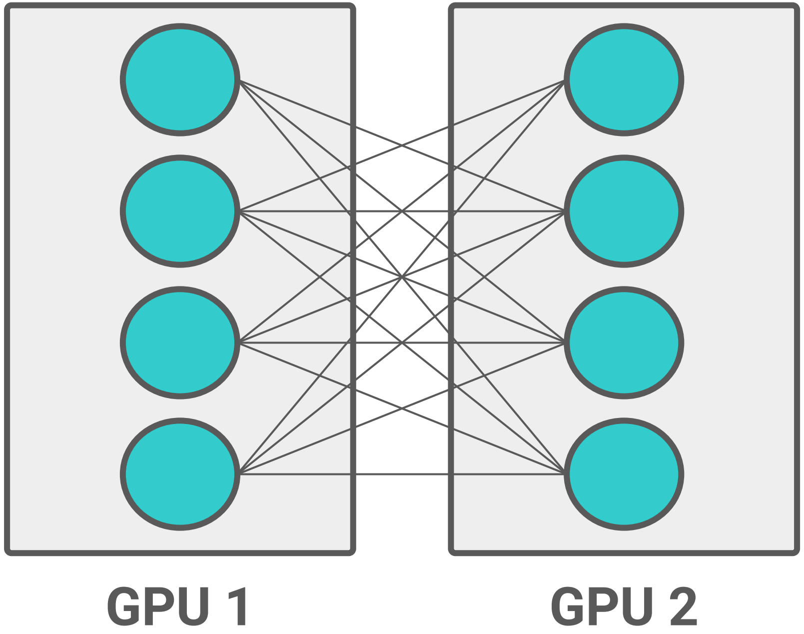
Model Parallelism (Large model training)

split the model across devices

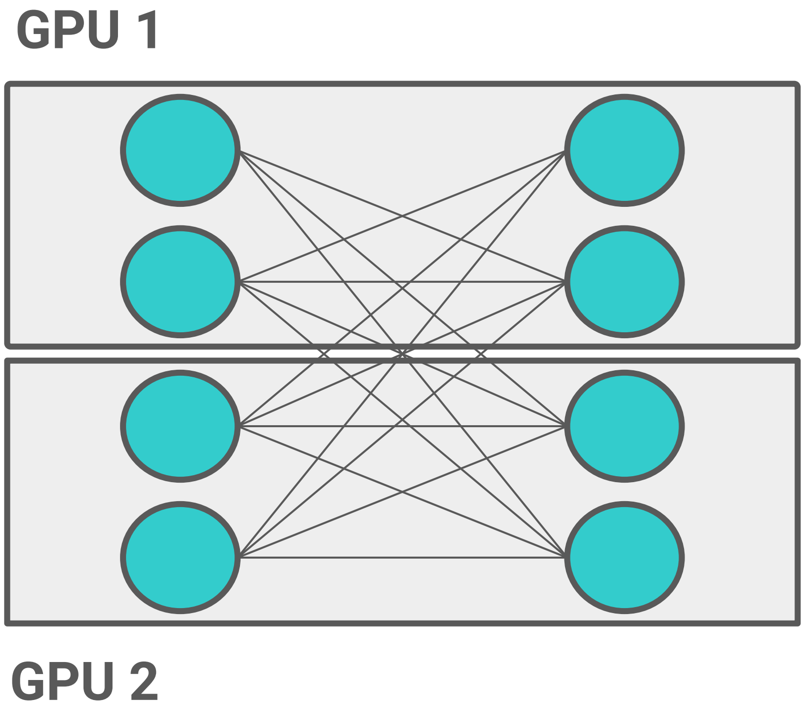
each device runs a fragment of the model



Pipeline Parallelism

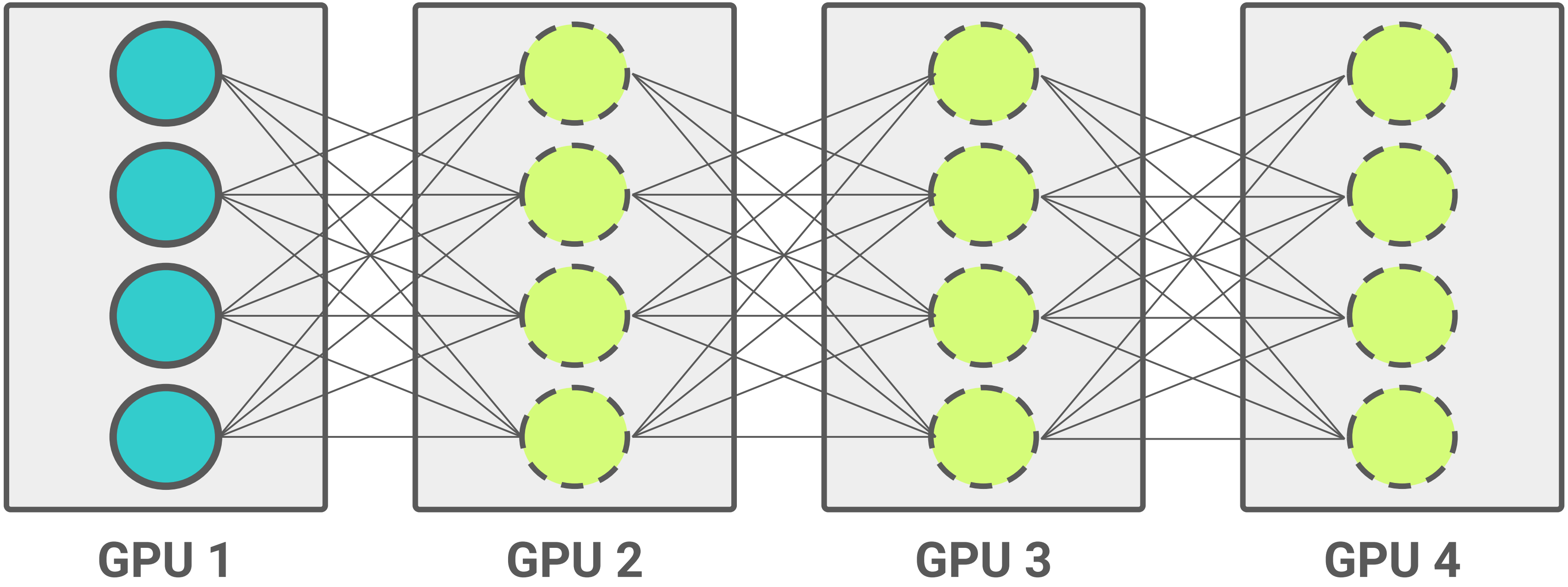


Distributed Tensor Computation



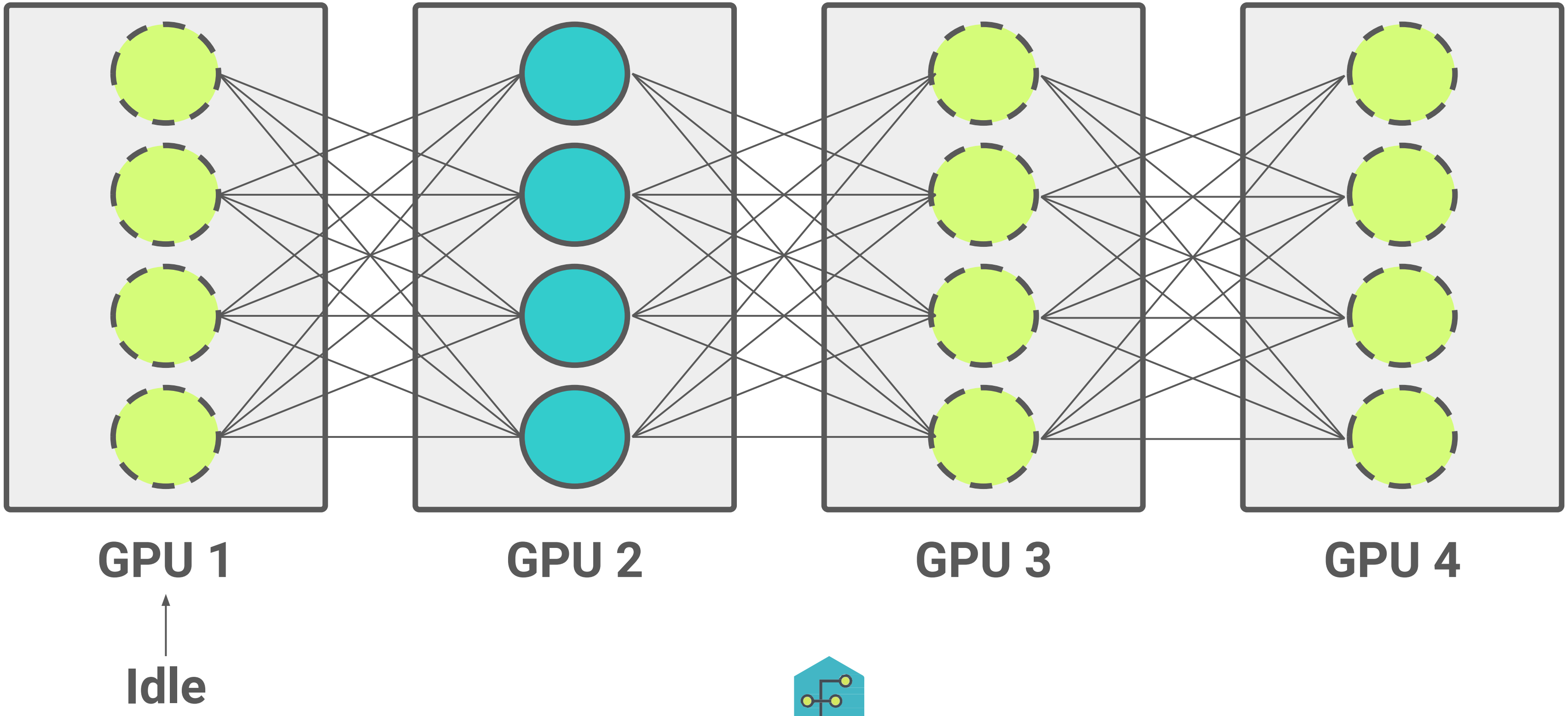
Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

Model Parallelism: Naive

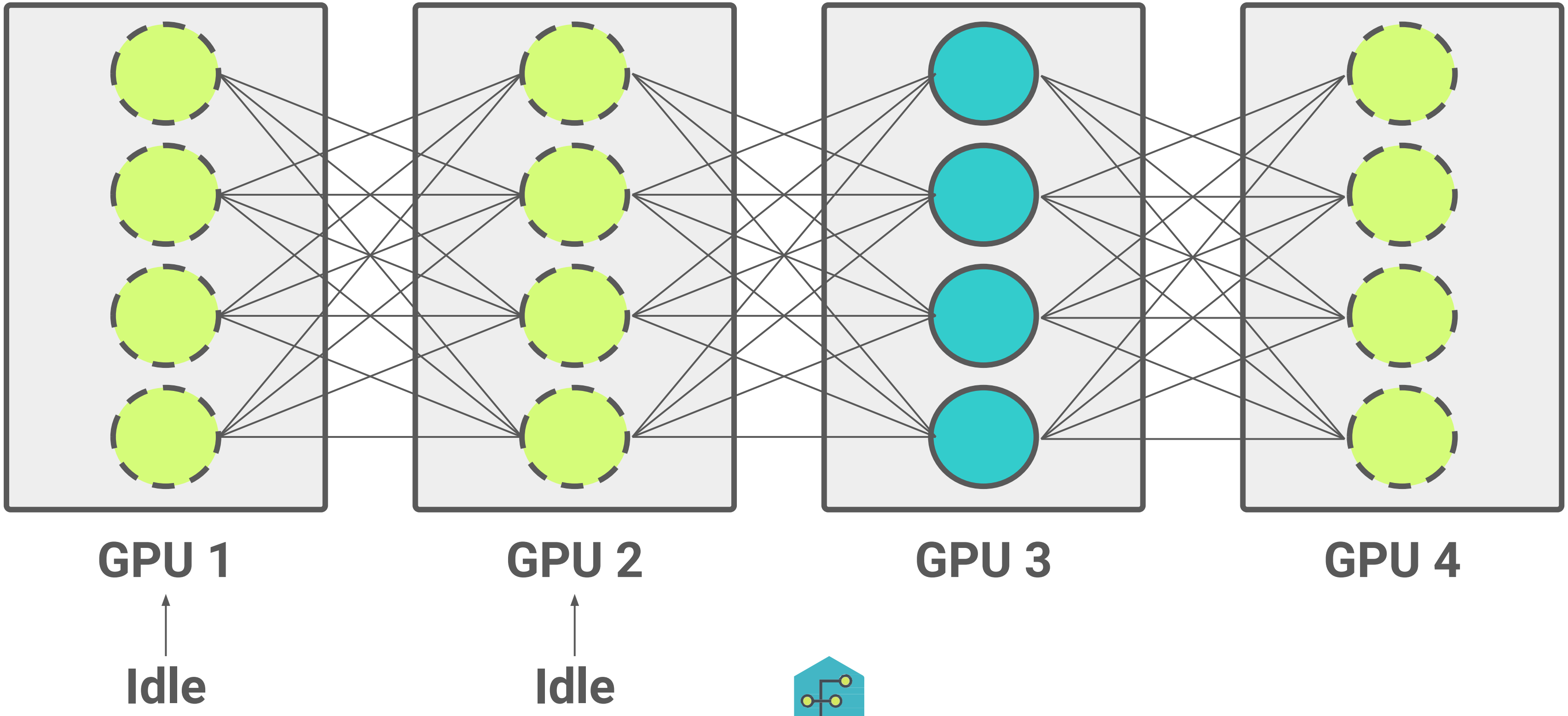


Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

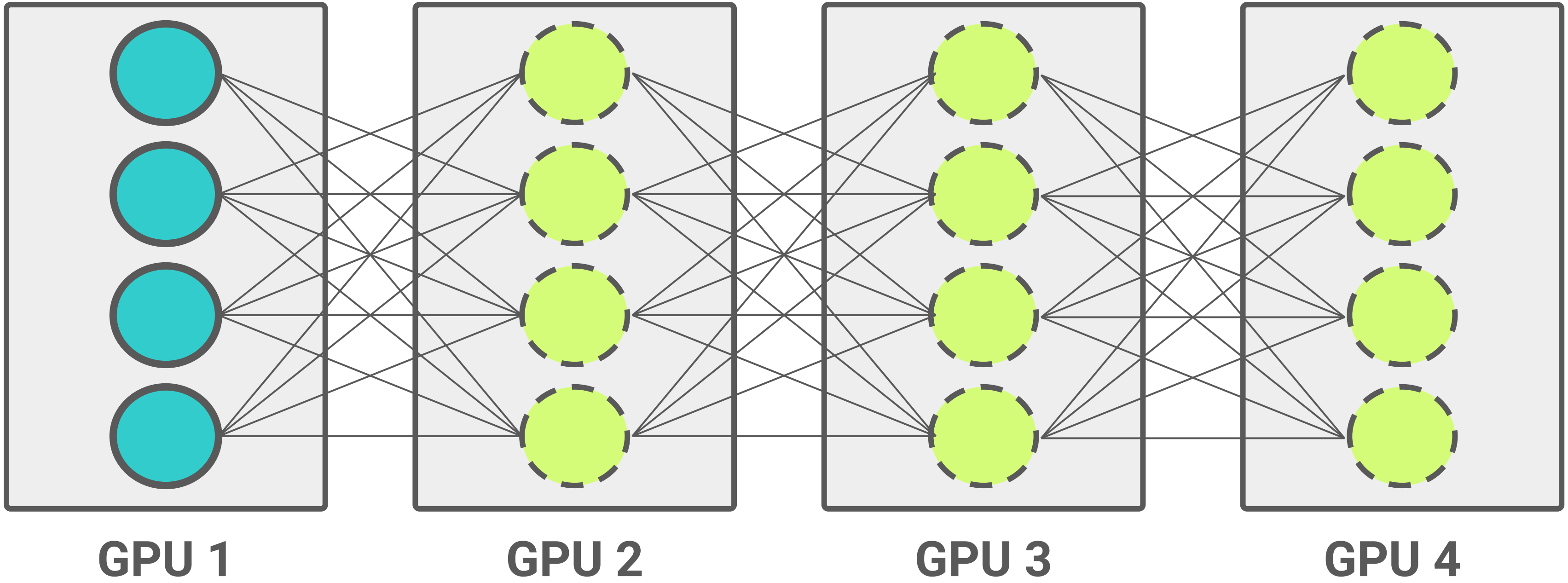
Model Parallelism: Naive



Model Parallelism: Naive

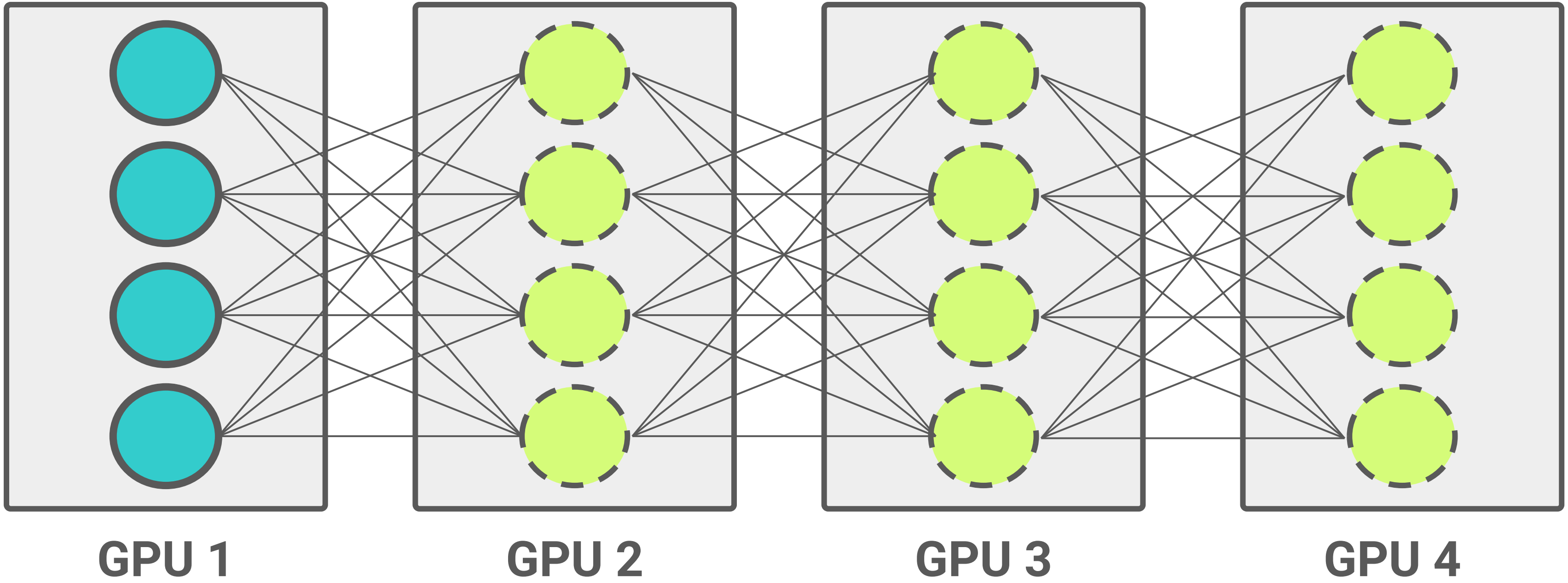


Pipeline Parallelism



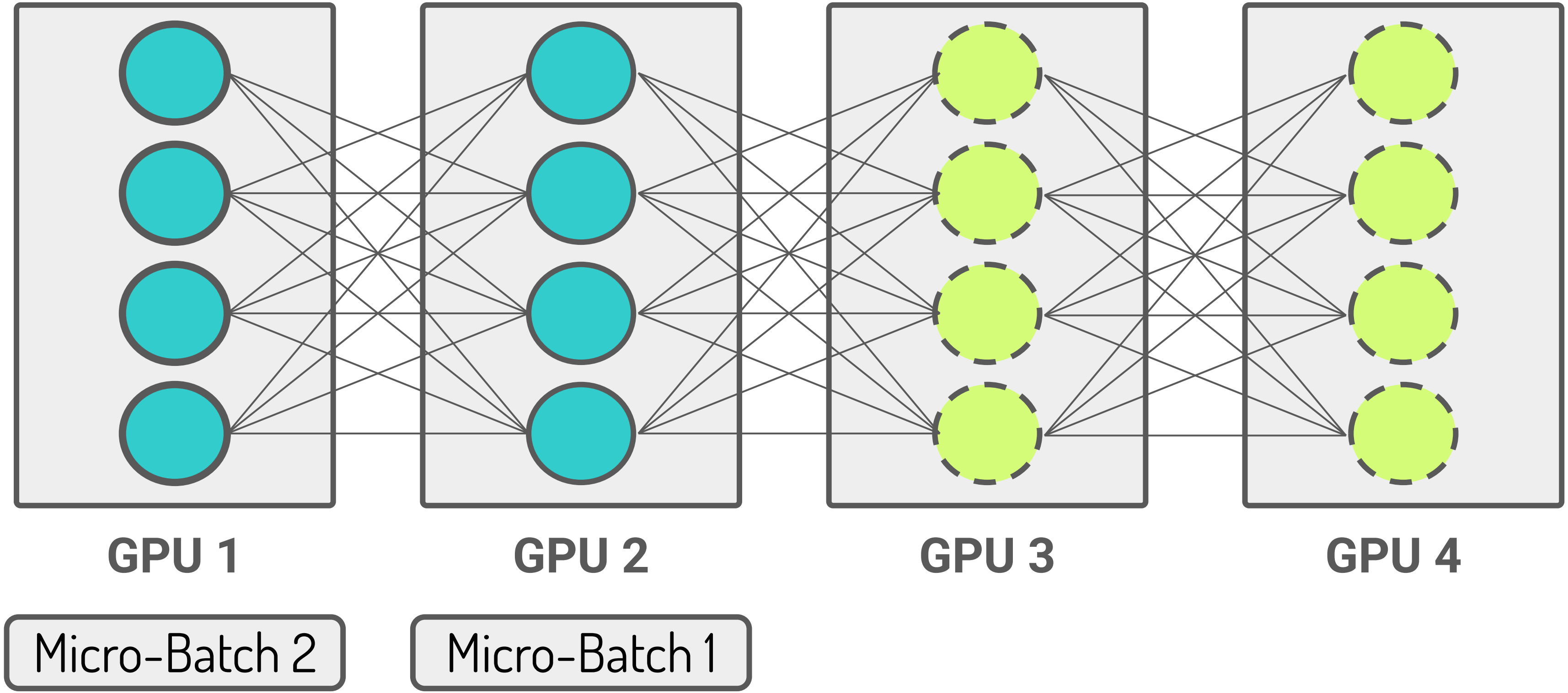
key idea: split mini-batch into sequential micro-batches

Pipeline Parallelism

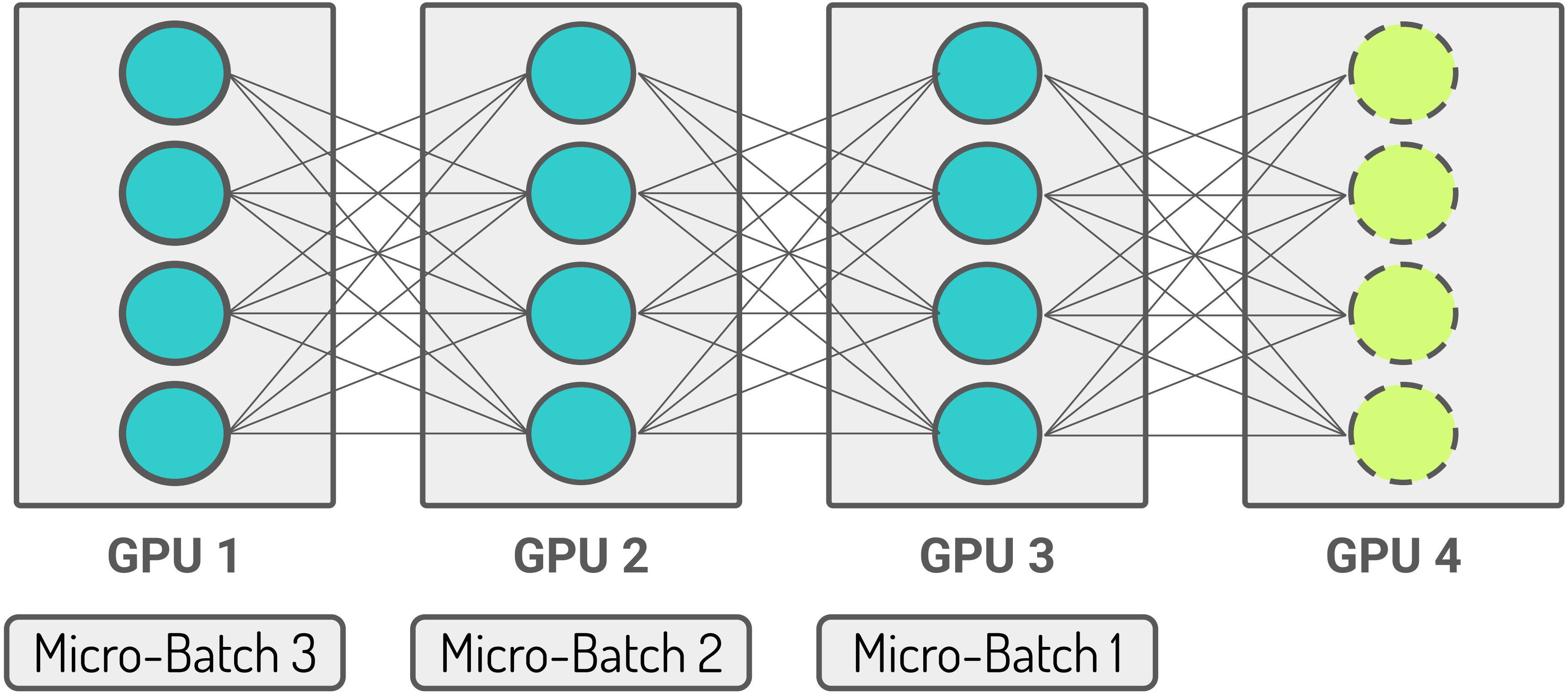


Micro-Batch 1

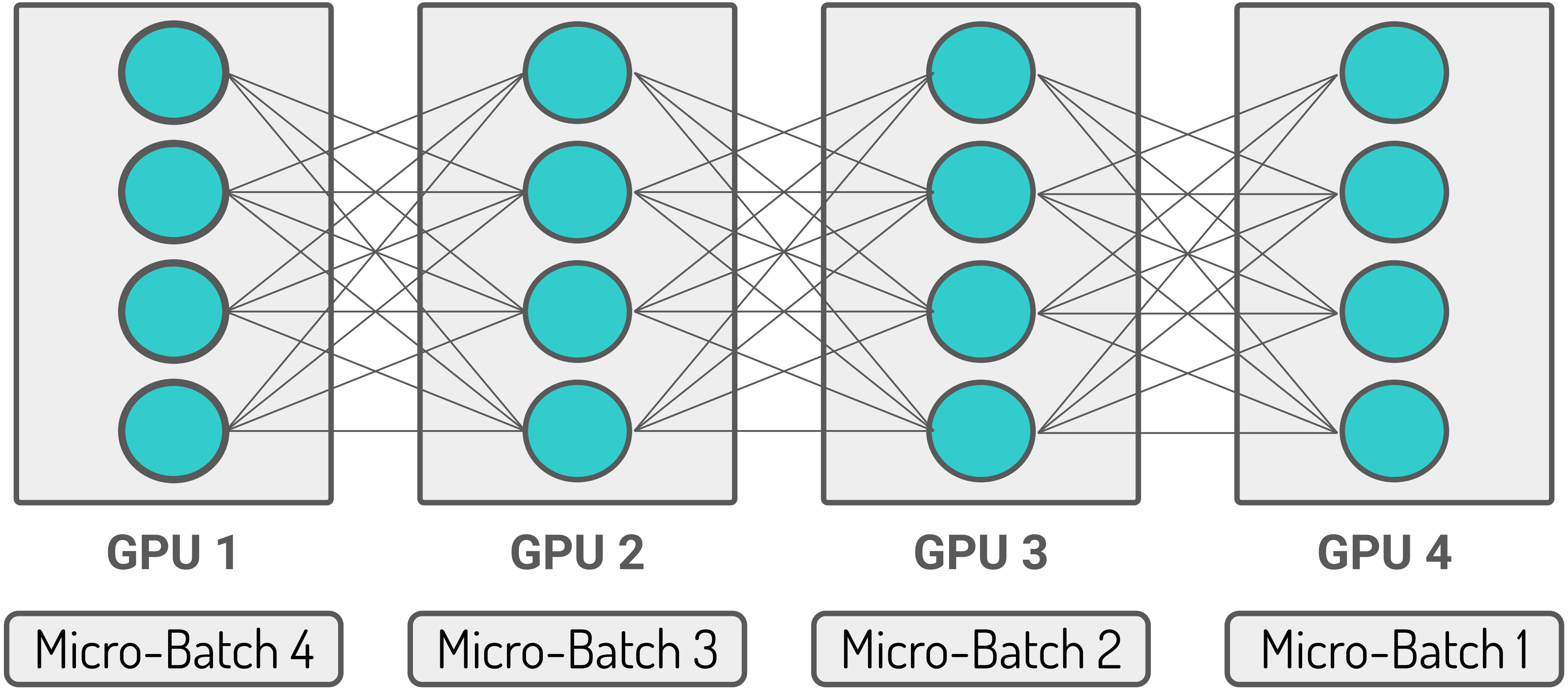
Pipeline Parallelism



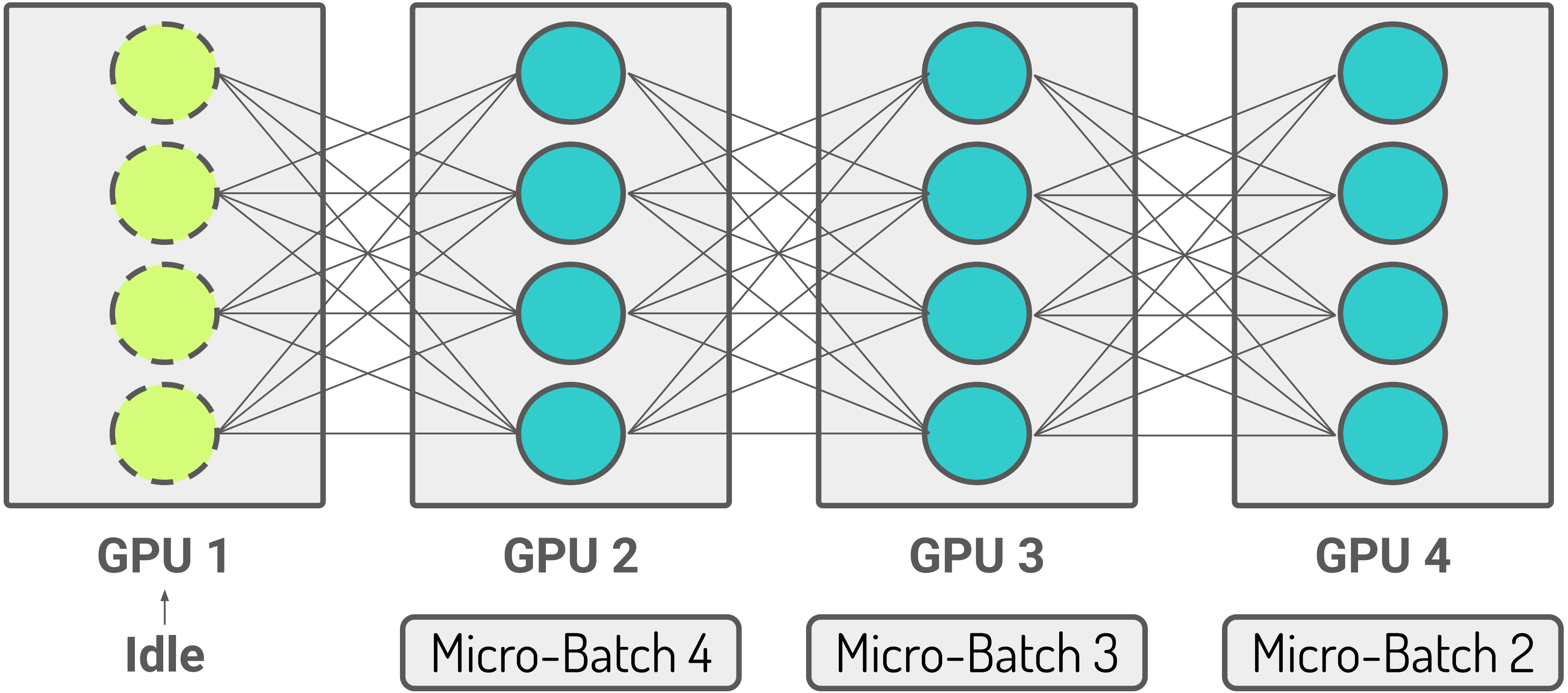
Pipeline Parallelism



Pipeline Parallelism



Pipeline Parallelism



Pros and cons of model parallelism

Pros:

- Can train bigger models.
- Implemented on Pytorch.

Cons:

- Not found in the distribution strategy of default libraries such as Tensorflow. (Mesh Tensorflow)
- Tricky to design an implement.

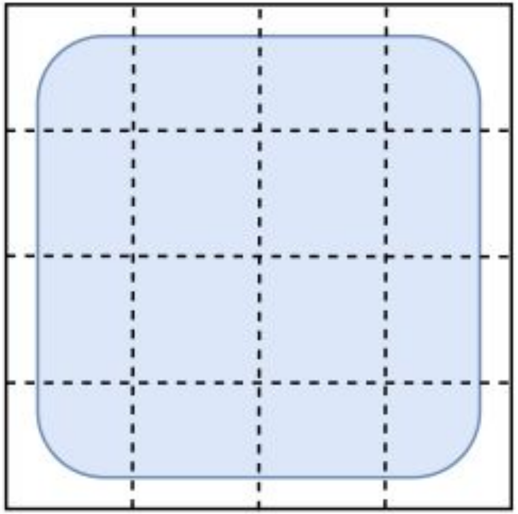
Data Parallelism (Large Batch Training)

split the data across devices

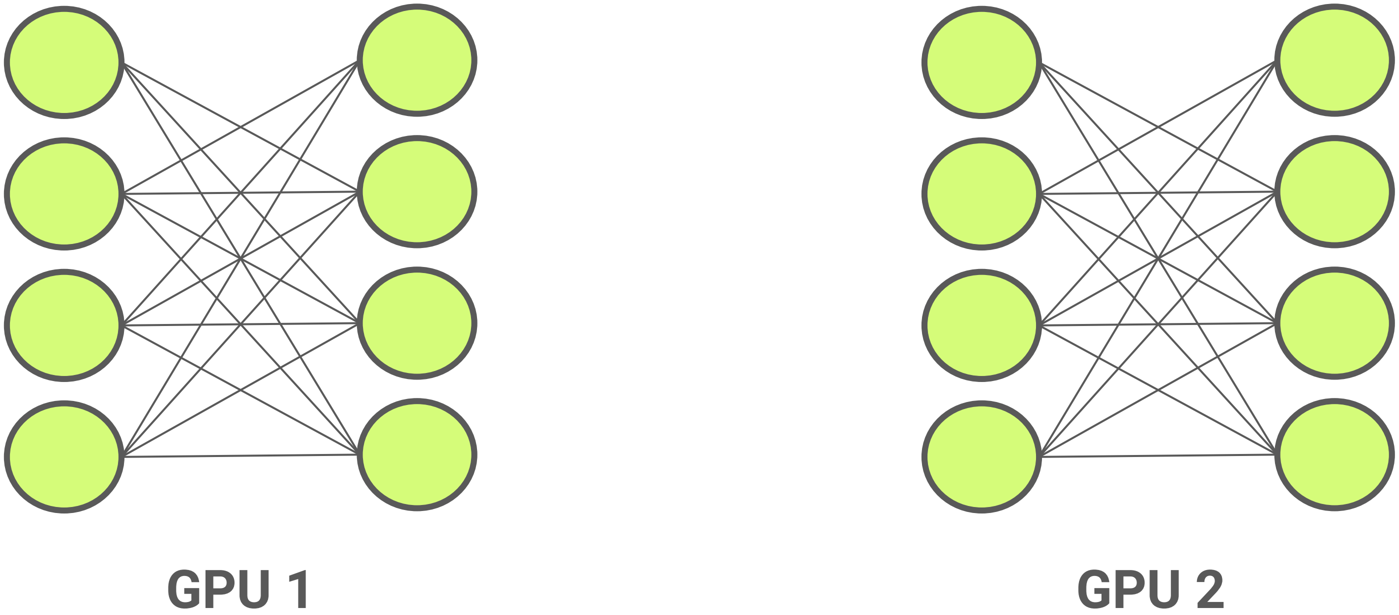
each device sees a fraction of the batch

each device replicates the model

each device replicates the optimizer



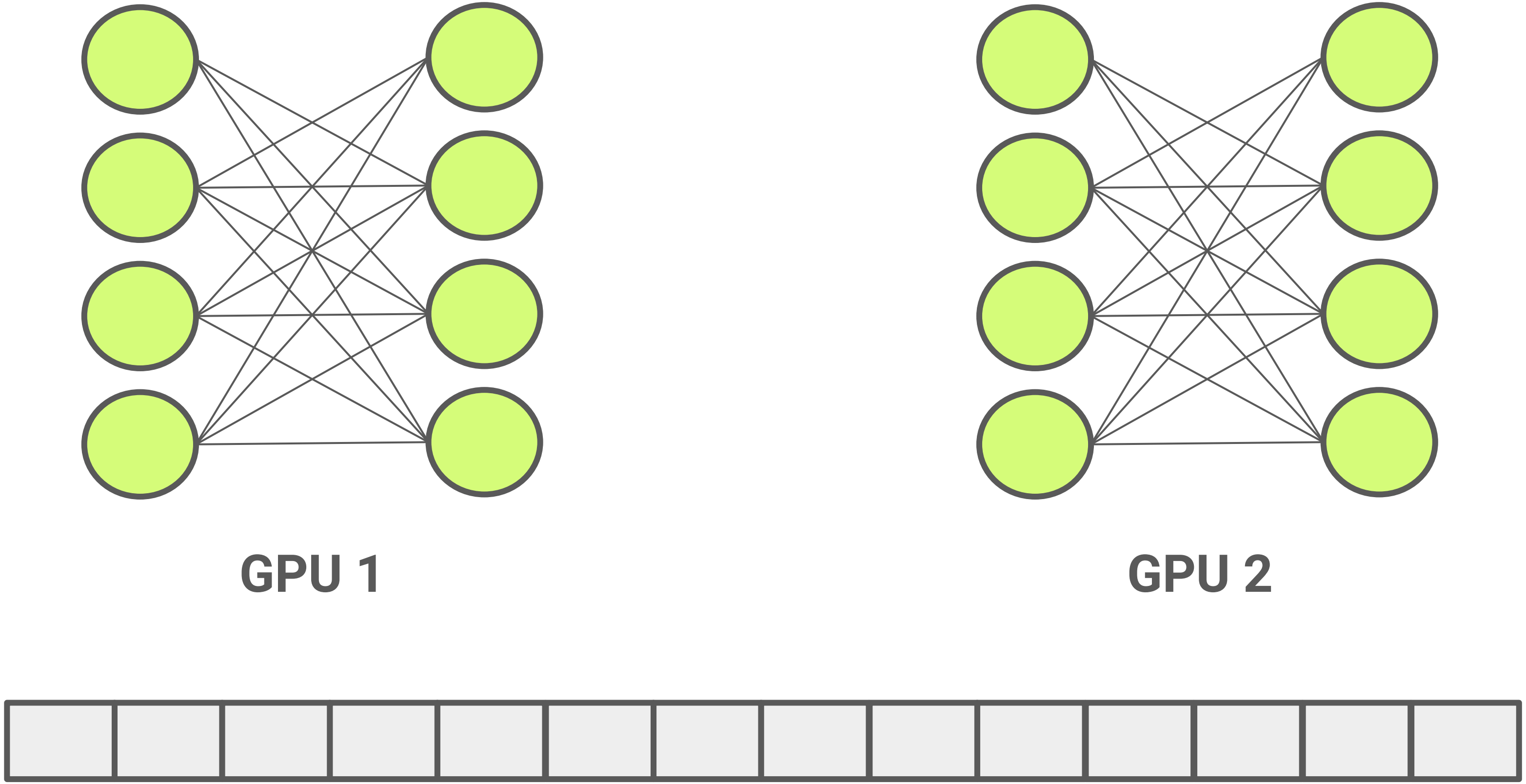
Data Parallelism



GPUs could be on same or multiple nodes

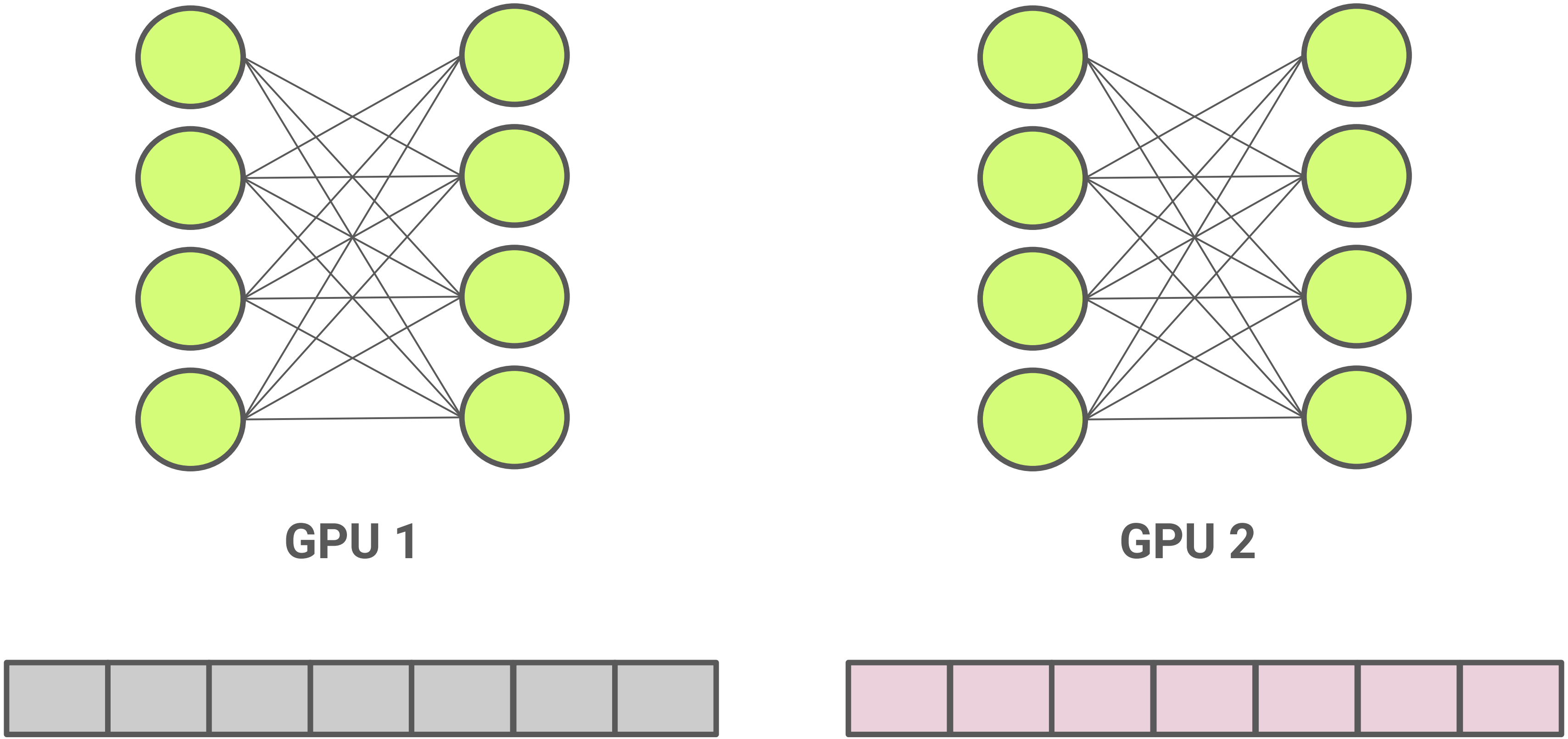
Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

Get a batch of data



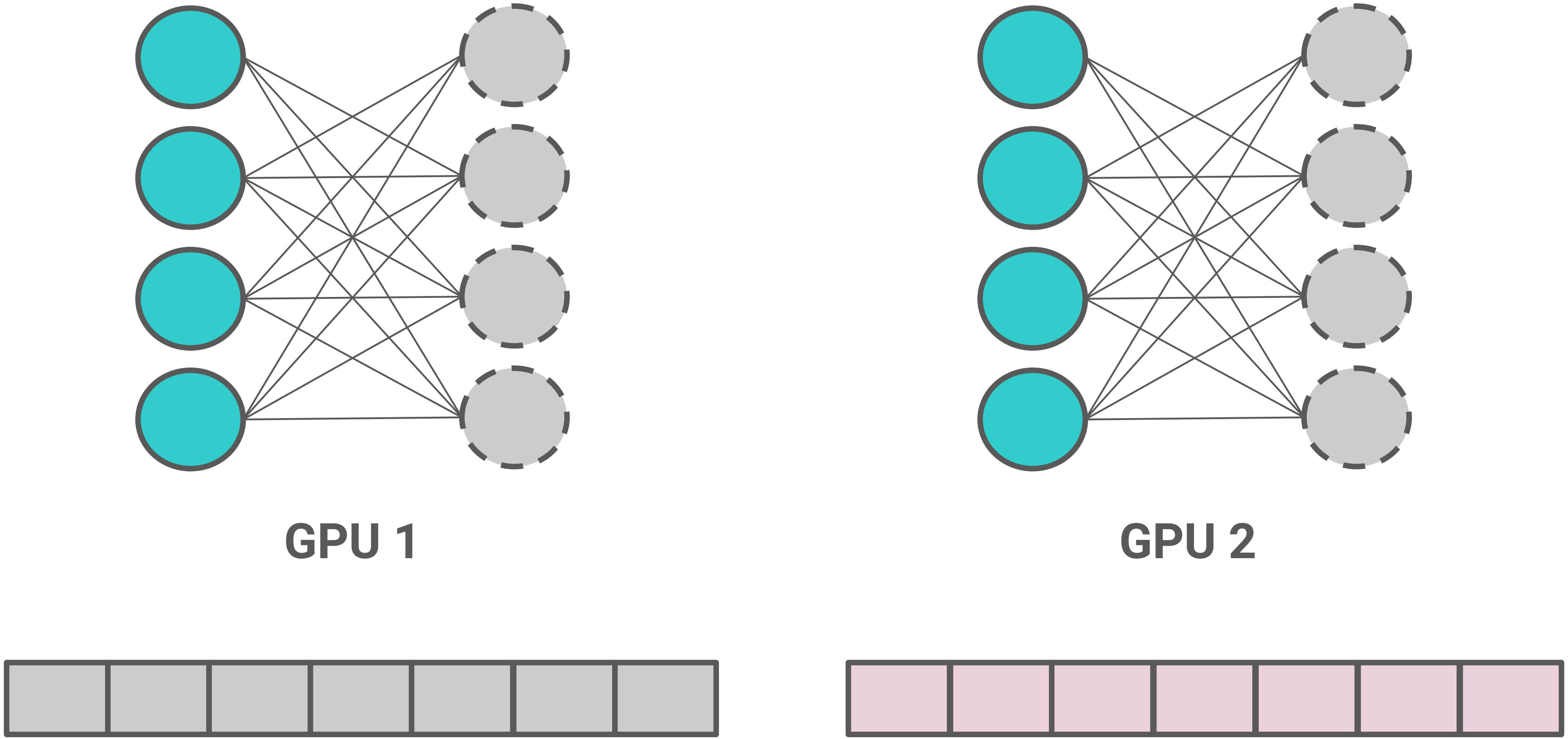
Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

Split batch across devices



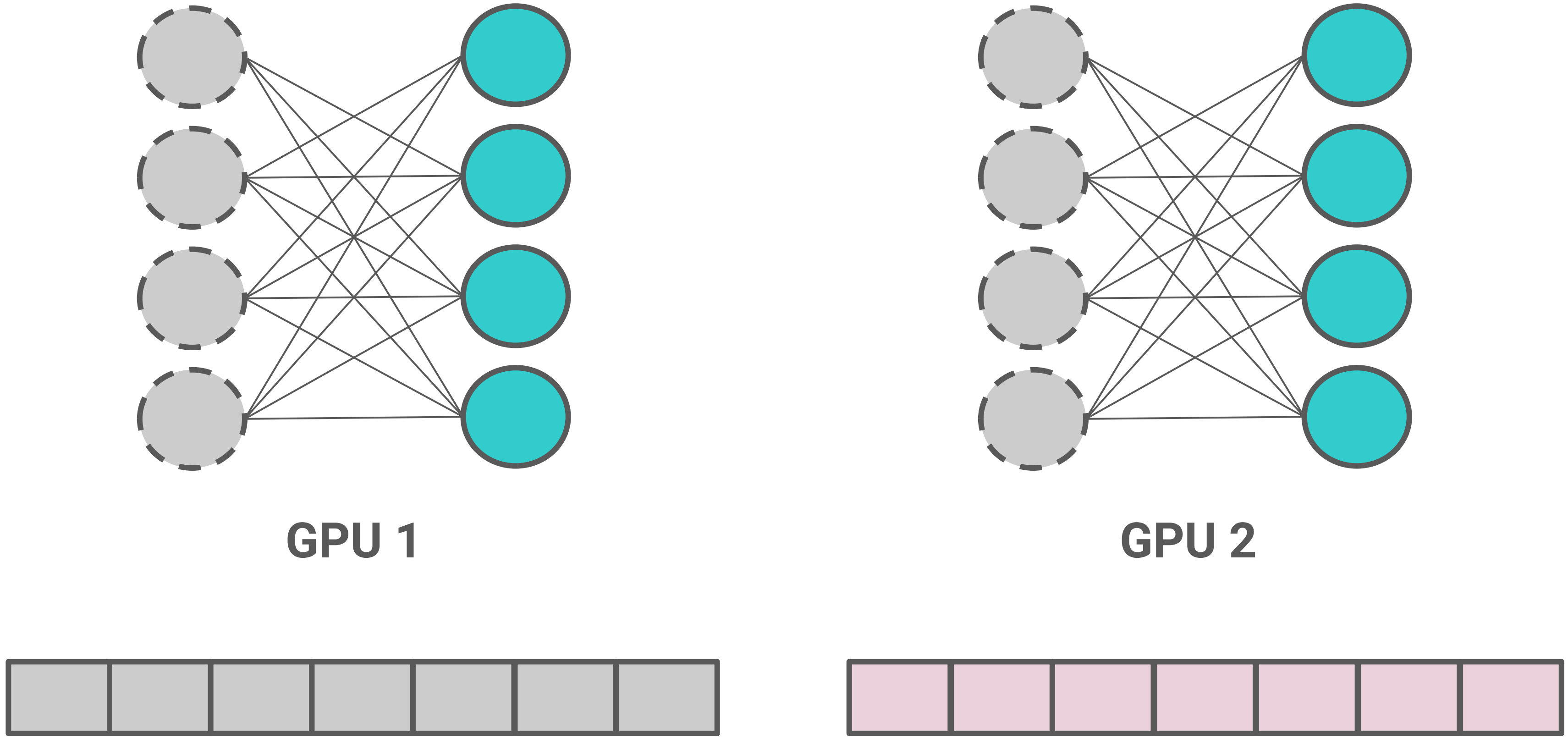
Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

Parallel forward passes



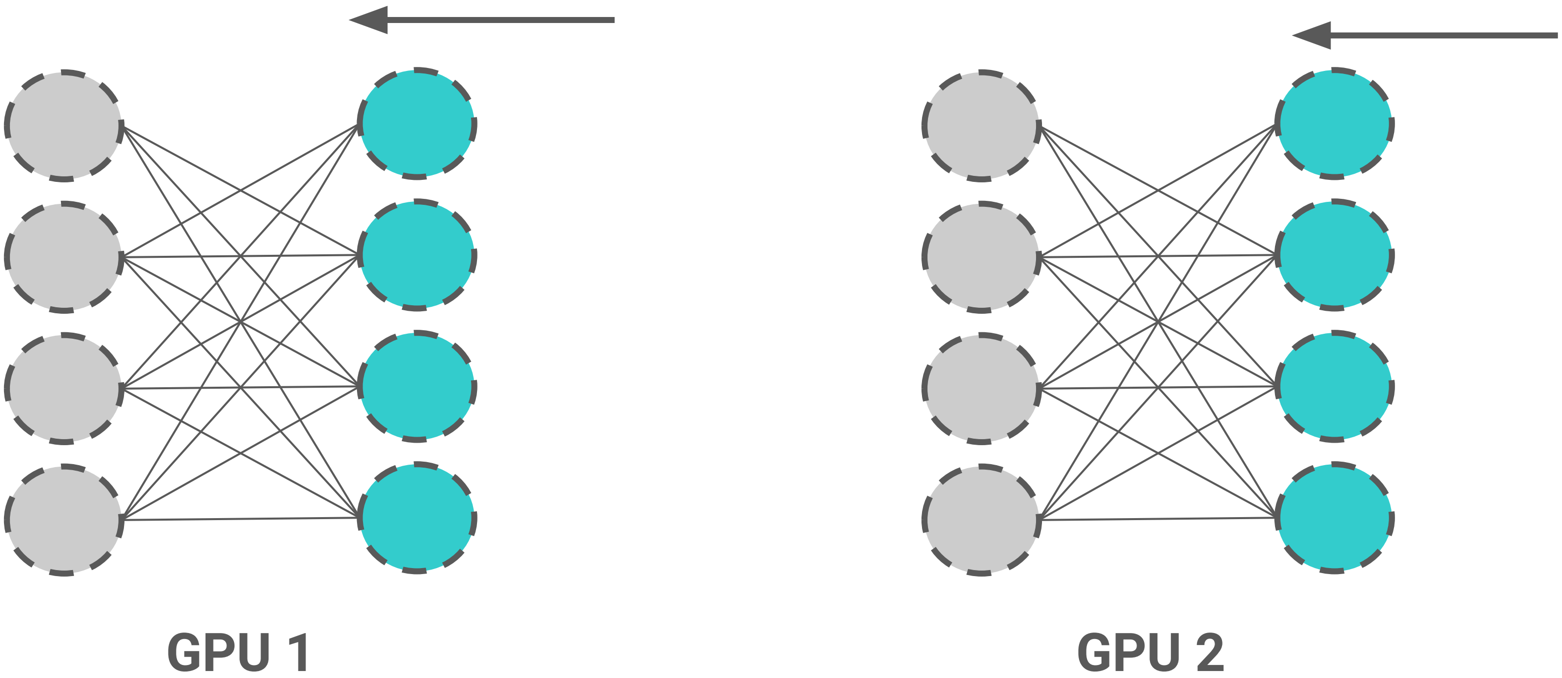
Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

Parallel forward passes



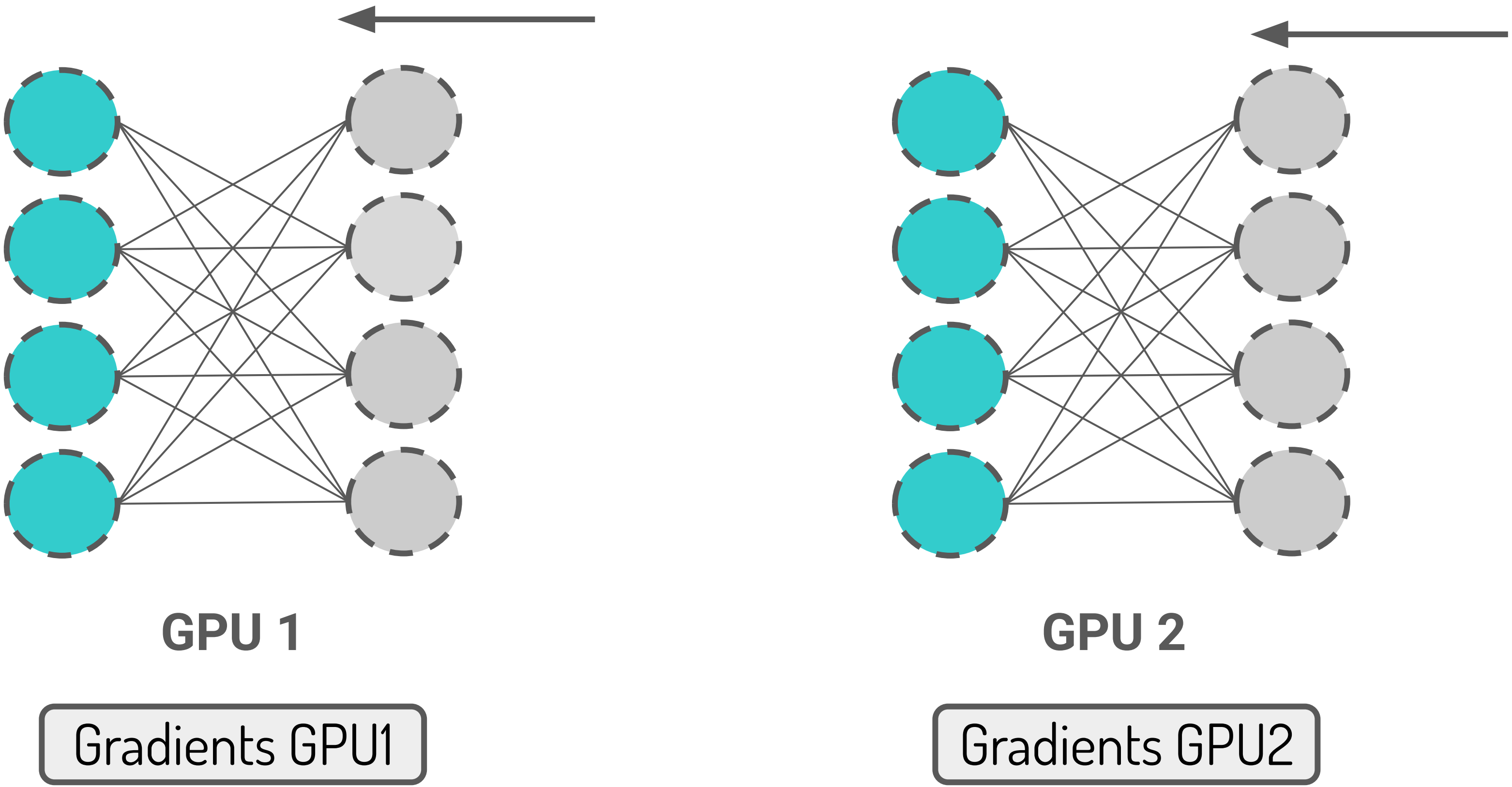
Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

Backpropagate gradients

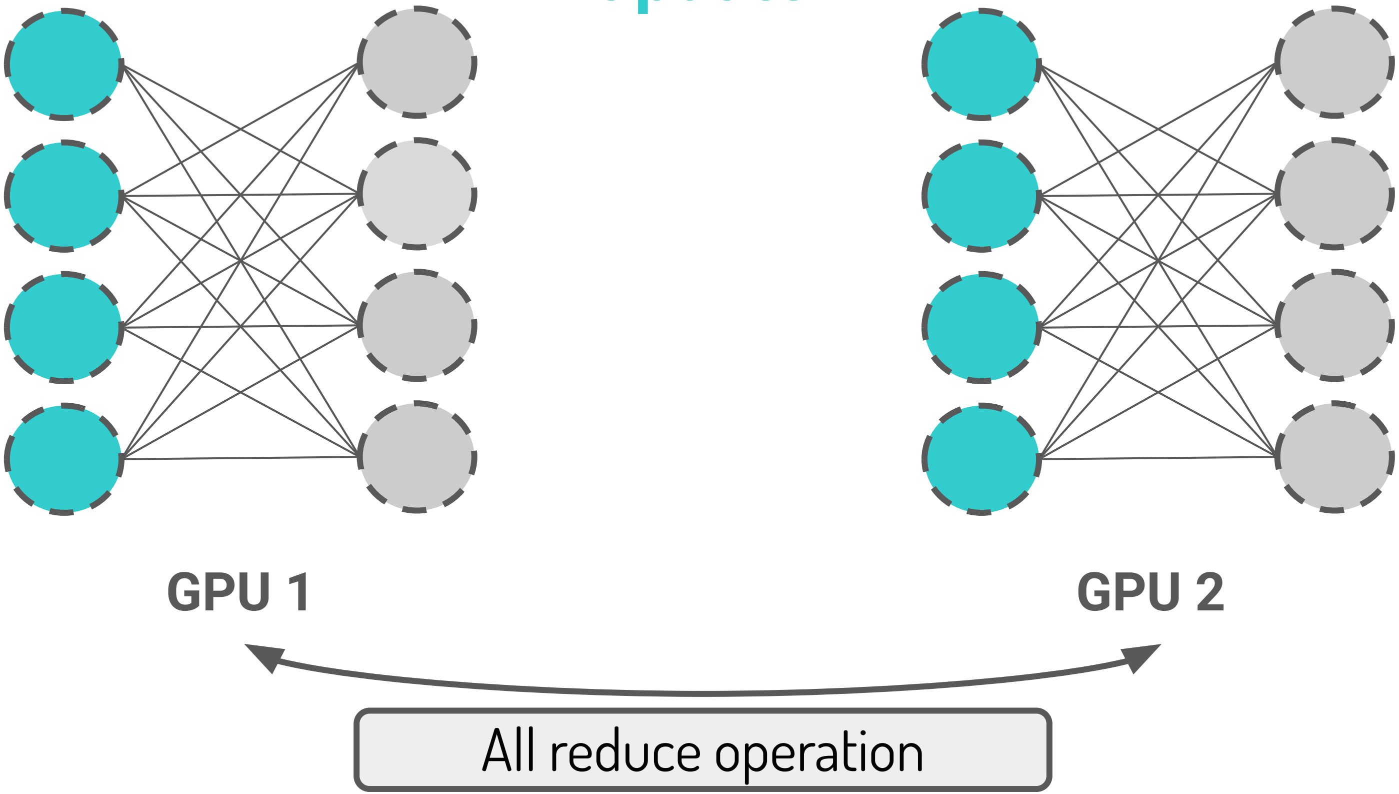


Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

Backpropagate gradients

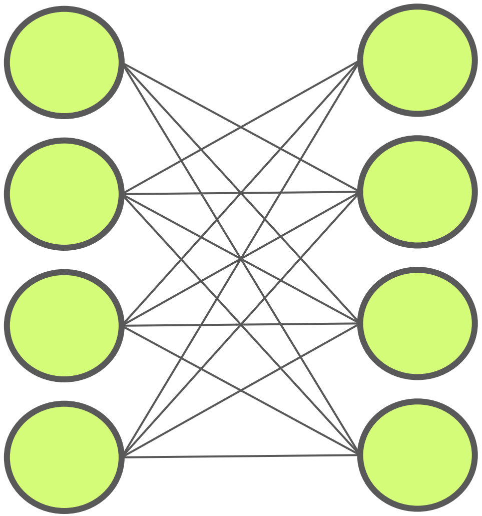


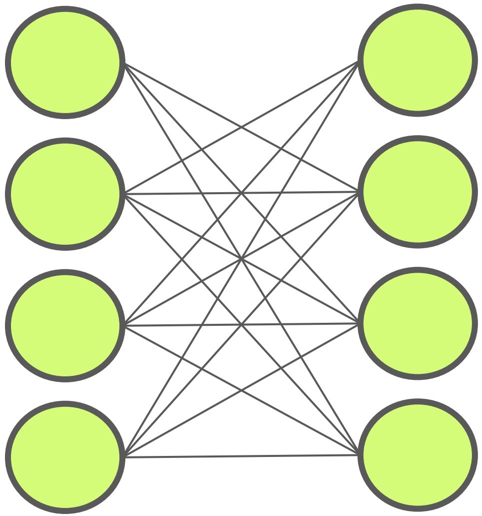
Share gradients among GPUs and update



Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

Start the next step with a new minibatch of data

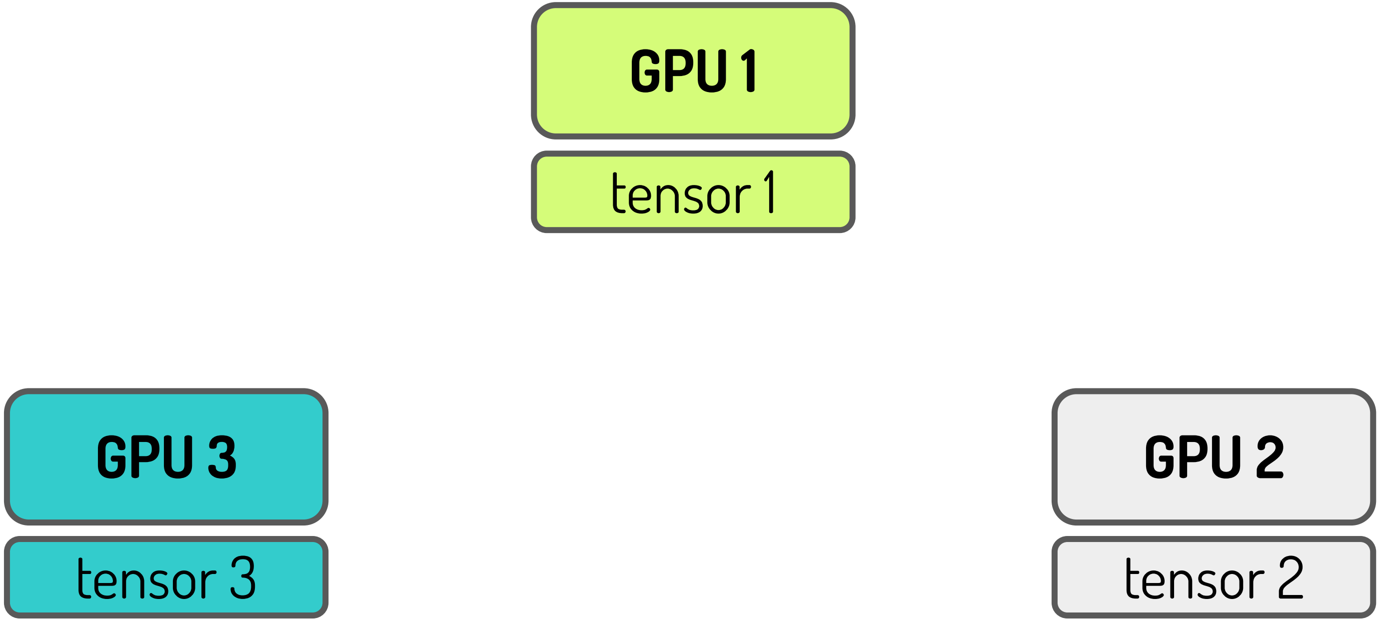




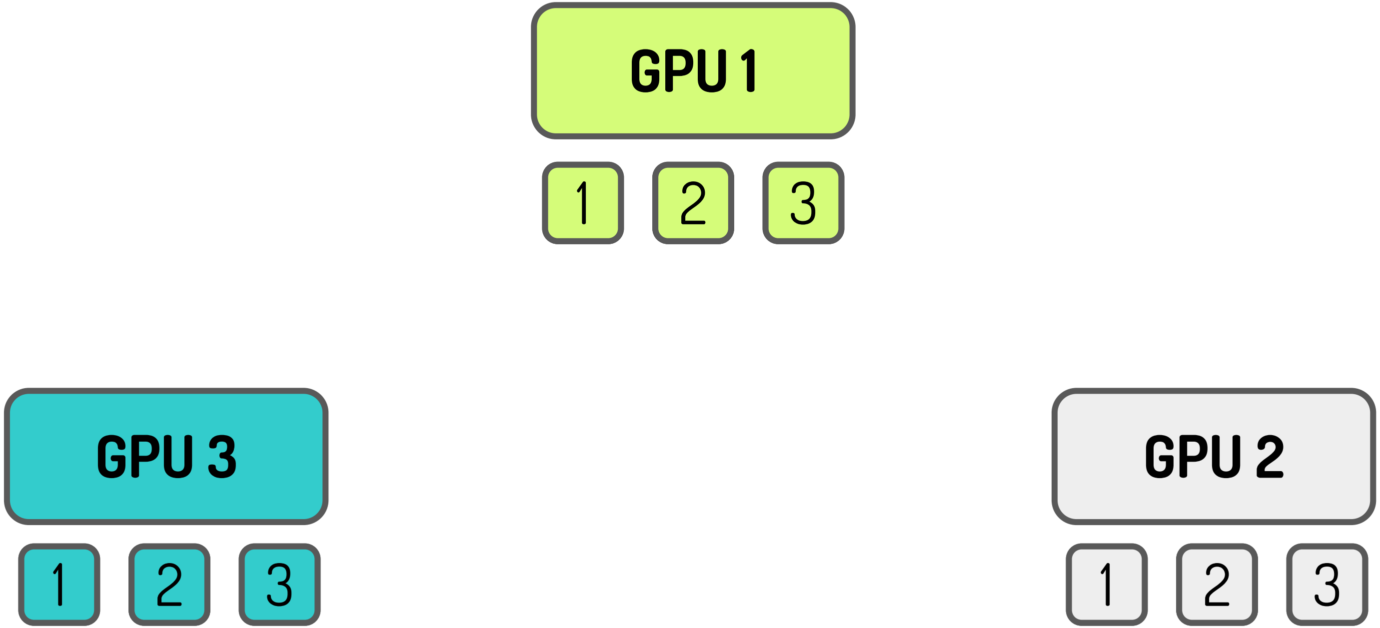
all parameters stay synchronized!

Credit: Scaling Laws for Neural Language Models, stanford cs329 slides

So, what's All-Reduce?

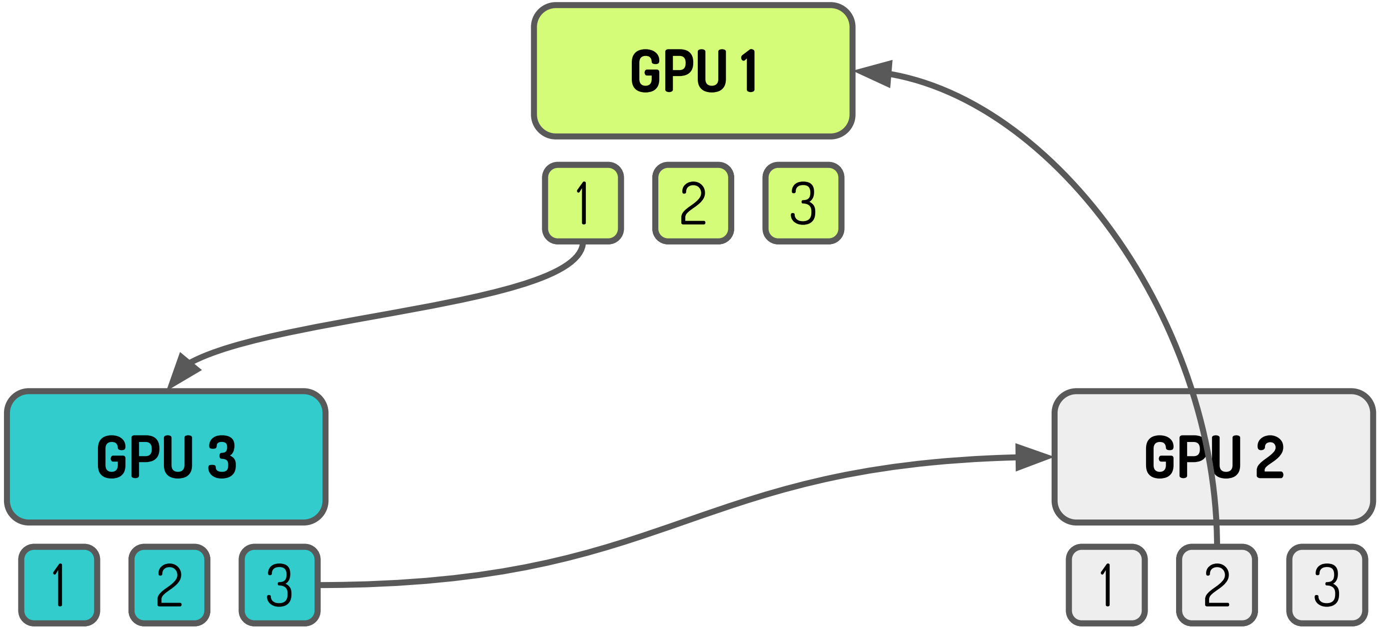


Ring All-Reduce



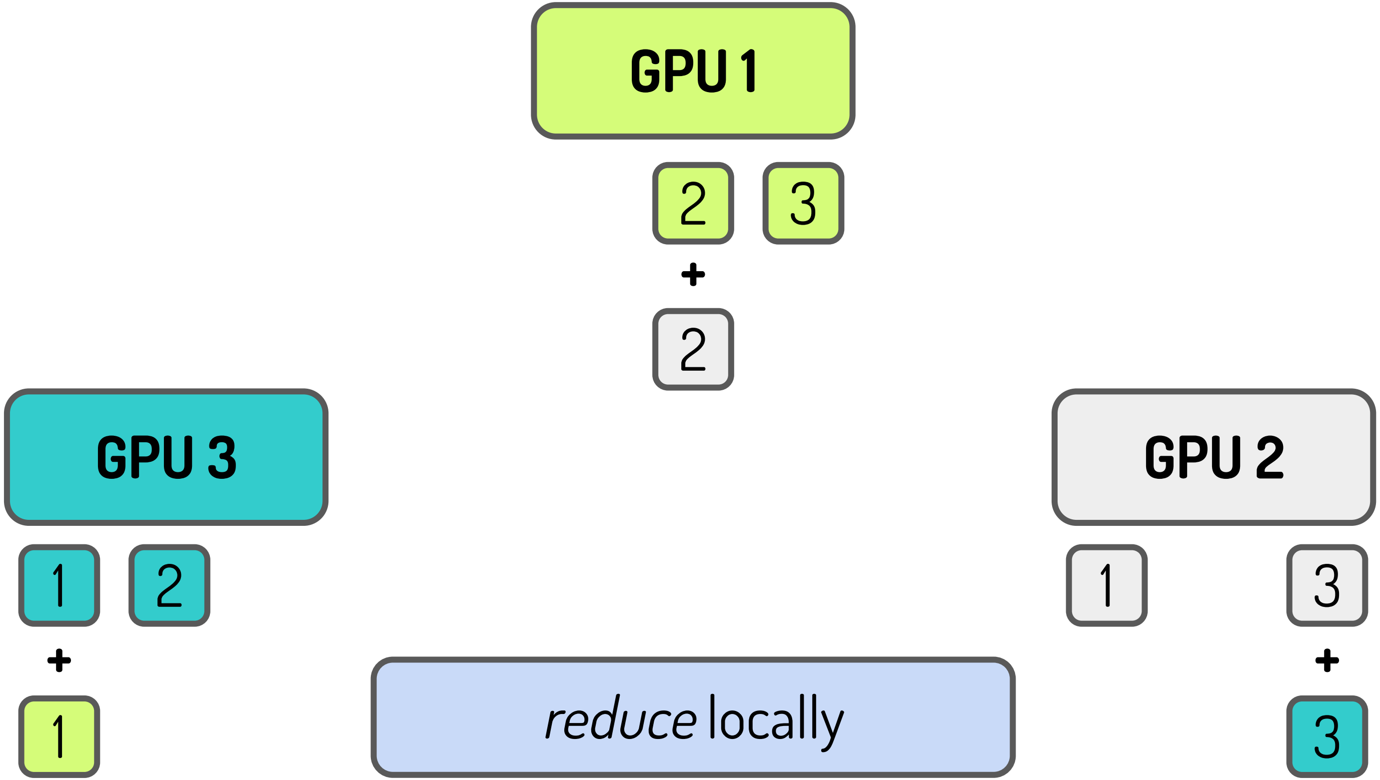
split tensor into p chunks

Ring All-Reduce

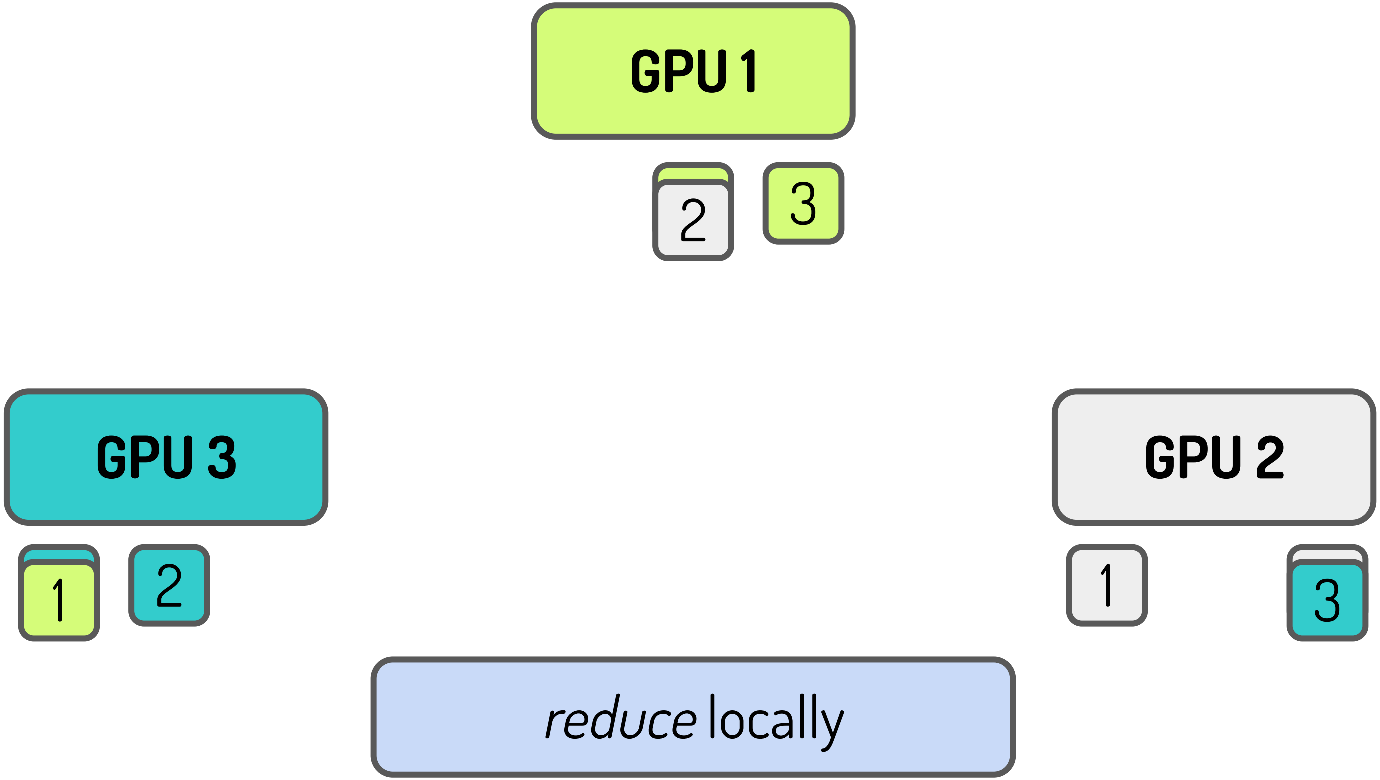


share a chunk

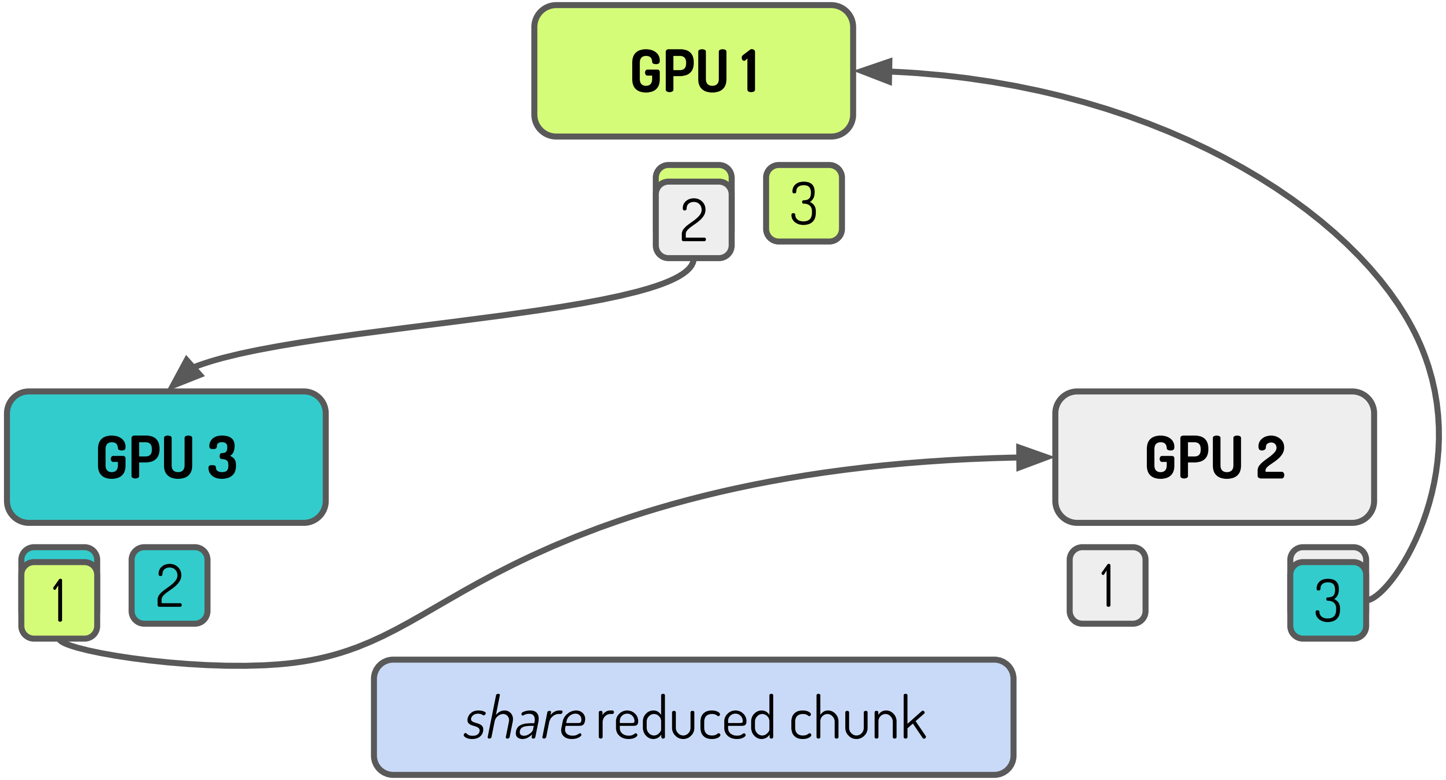
Ring All-Reduce



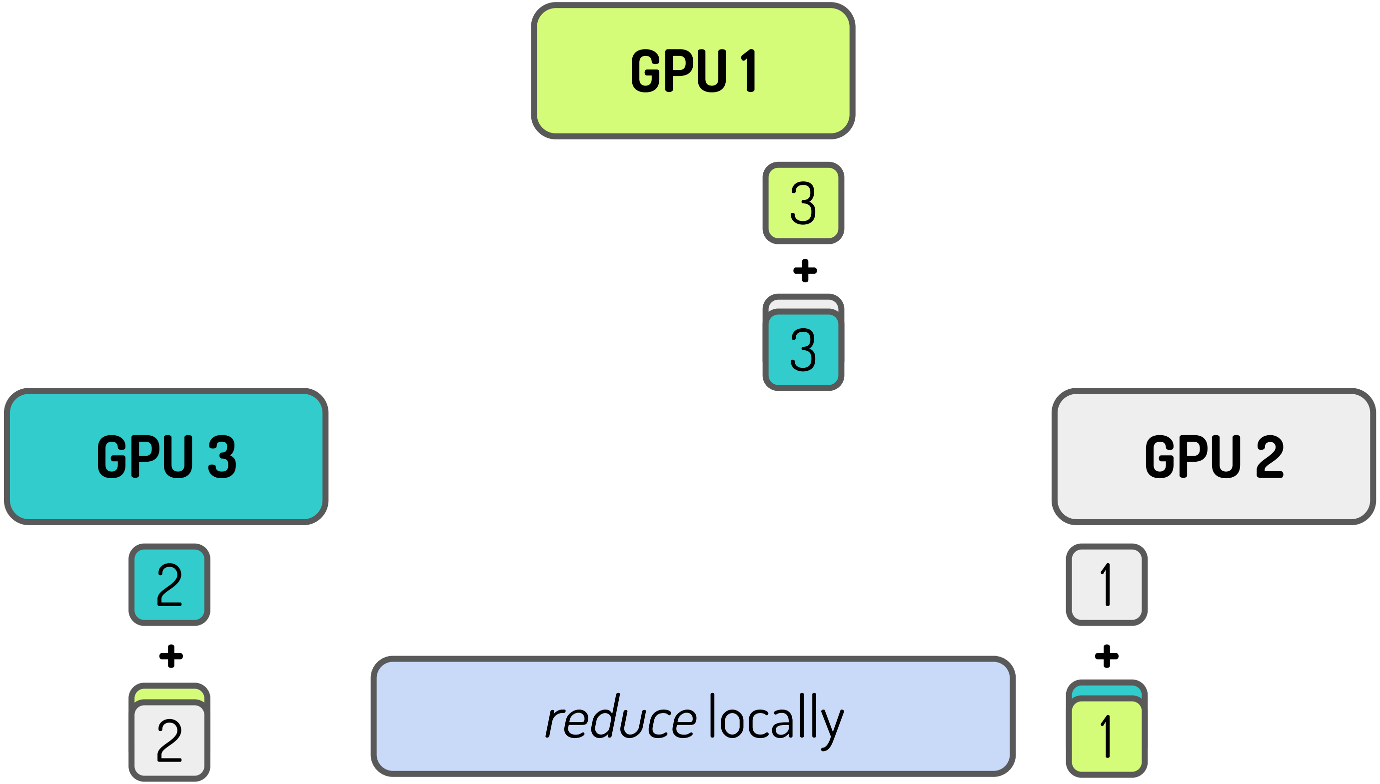
Ring All-Reduce



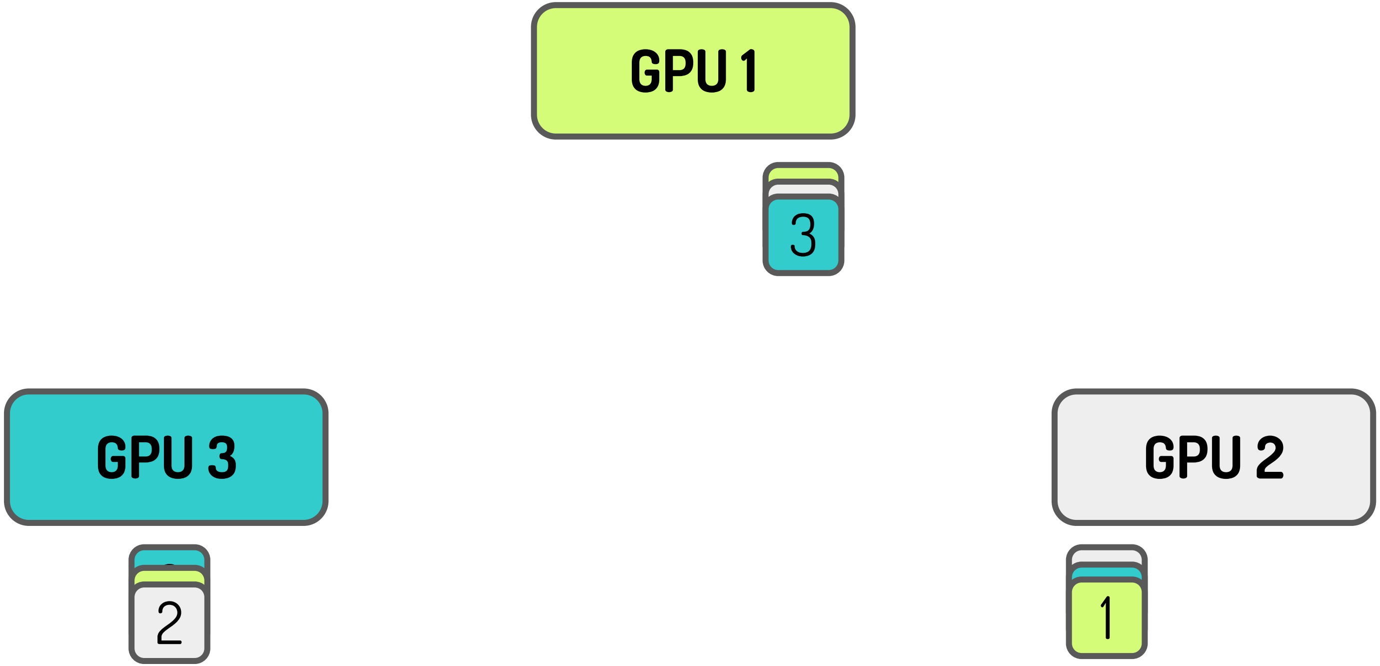
Ring All-Reduce



Ring All-Reduce

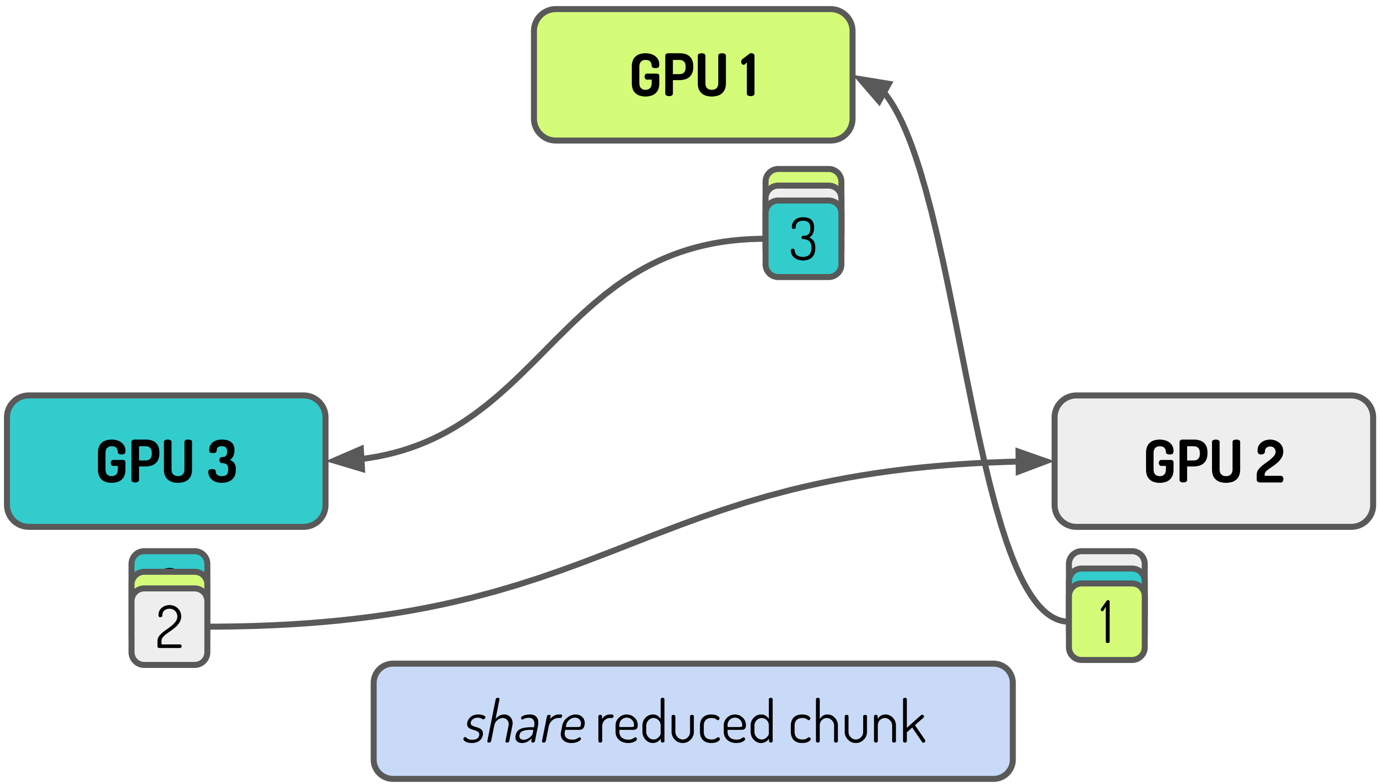


Ring All-Reduce

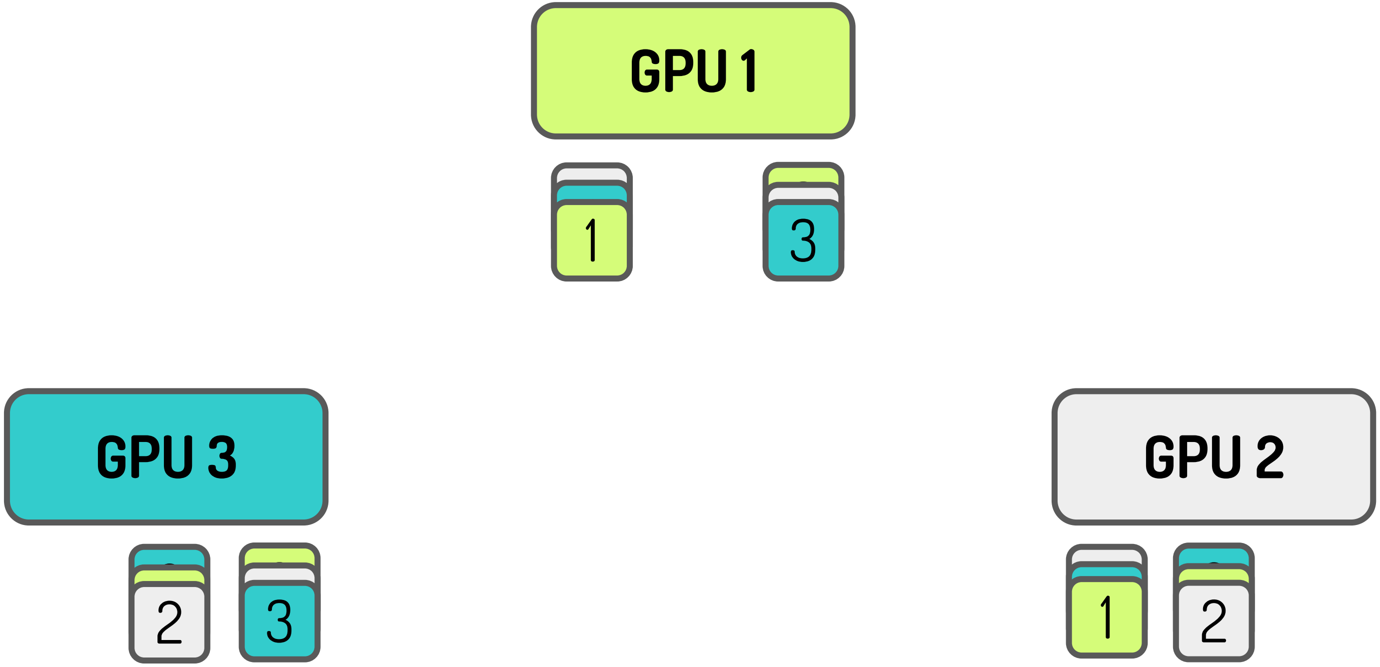


everyone has a chunk of the result after *(p-1)* share-reduce steps

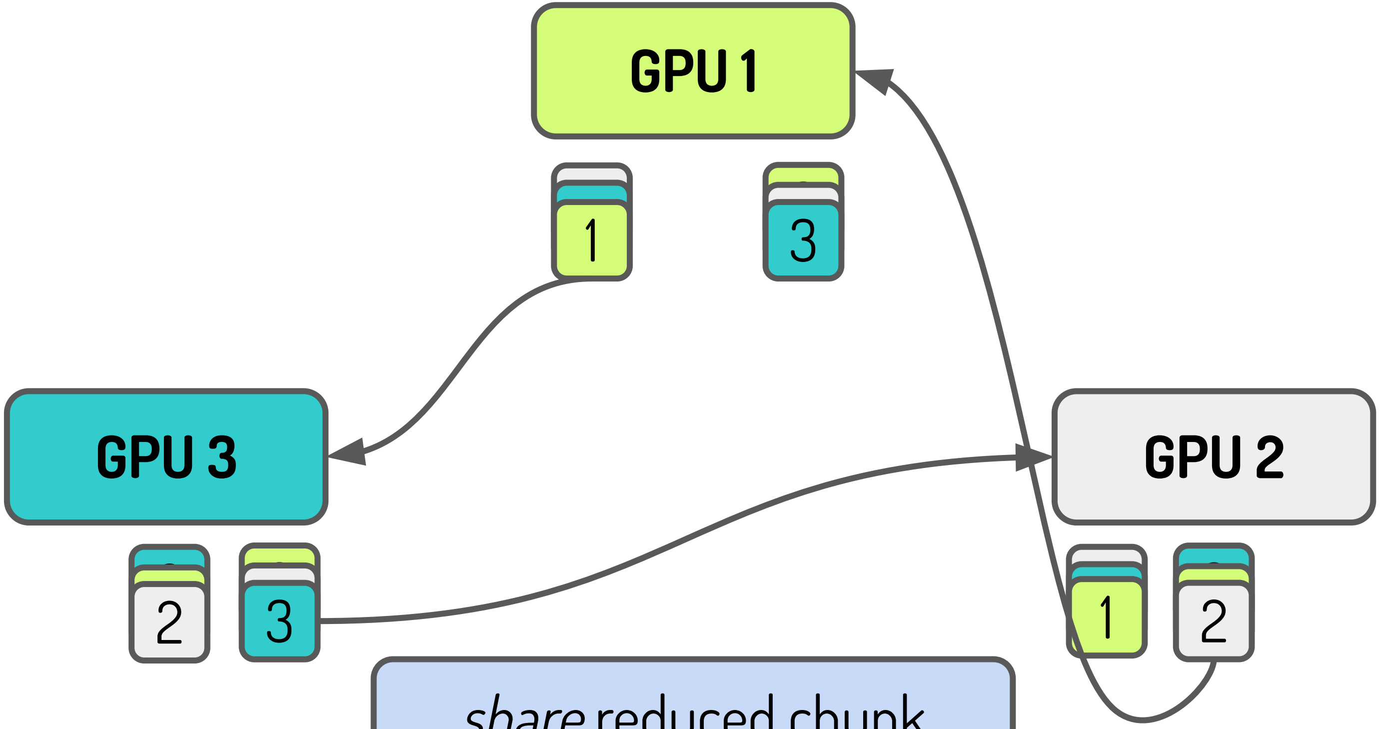
Ring All-Reduce



Ring All-Reduce

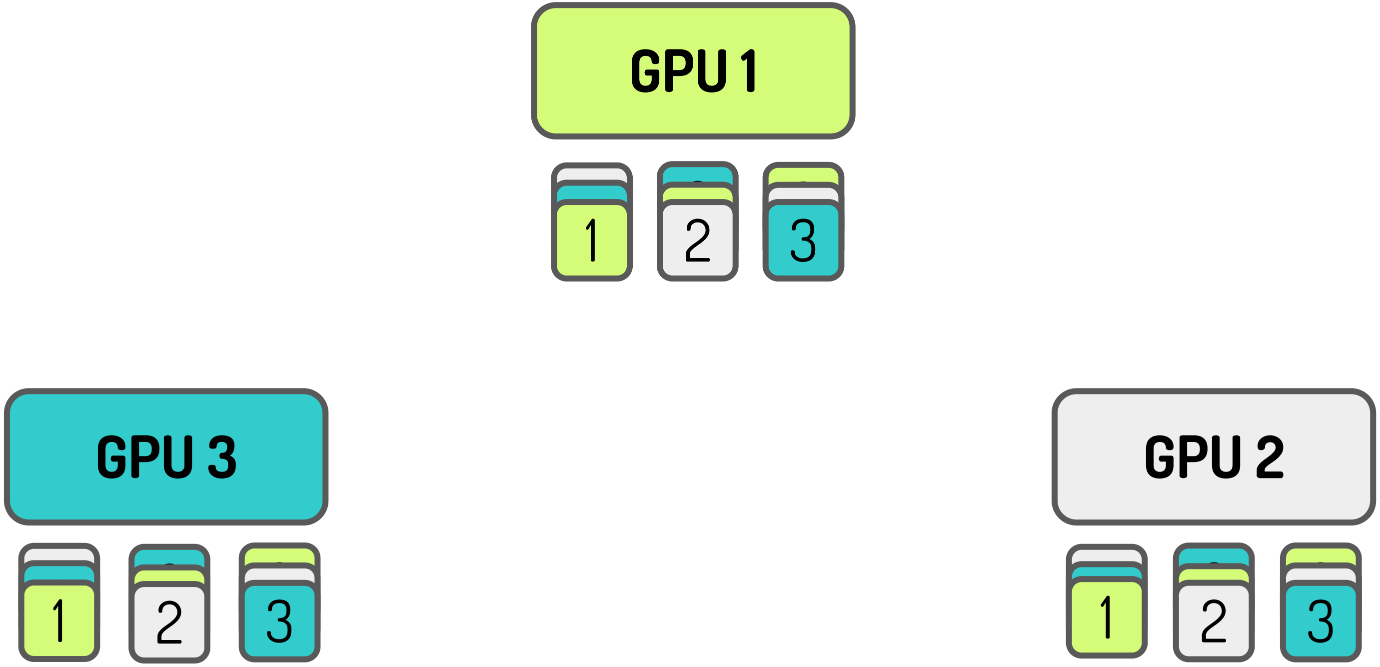


Ring All-Reduce



share reduced chunk

Ring All-Reduce



Ring All-Reduce Advantages

Naive method = p senders x (p - 1) receivers x o(n) tensor = **o(np^2)**
everyone does **o(np)** work.

Manager node method = (p-1) x 2 transfers x o(n) tensor = **o(np)**
manager does **o(np)** work

Ring All-Reduce = p senders x 1 receiver x o(n/p) tensor x (p-1) rounds x 2 phases = **o(np)**
everyone does <u>equal</u> **o(n)** work (independent of p)

Before the demo, what strategies are implemented in Tensorflow

Training API	MirroredStrategy	TPUStrategy	MultiWorkerMirroredStrategy	CentralStorageStrategy	ParameterServerStrategy
Keras API	Supported	Supported	Supported	Experimental support	Supported planned post 2.4
Custom training loop	Supported	Supported	Supported	Experimental support	Experimental support
Estimator API	Limited Support	Not supported	Limited Support	Limited Support	Limited Support

Demos

Using data parallelism with model.fit

Using data parallelism with a custom training loop.

(https://github.com/juanma9613/Reducing-deep-learnin
g-training-times-Pycon2021)

THANK YOU!

- https://www.linkedin.com/in/juan-manuel-munoz-betancur/
- jmunozb@eafit.edu.co

Link to demo:

- https://github.com/juanma9613/Reducing-deep-learning-training-times-Pycon2021

Factored is hiring!