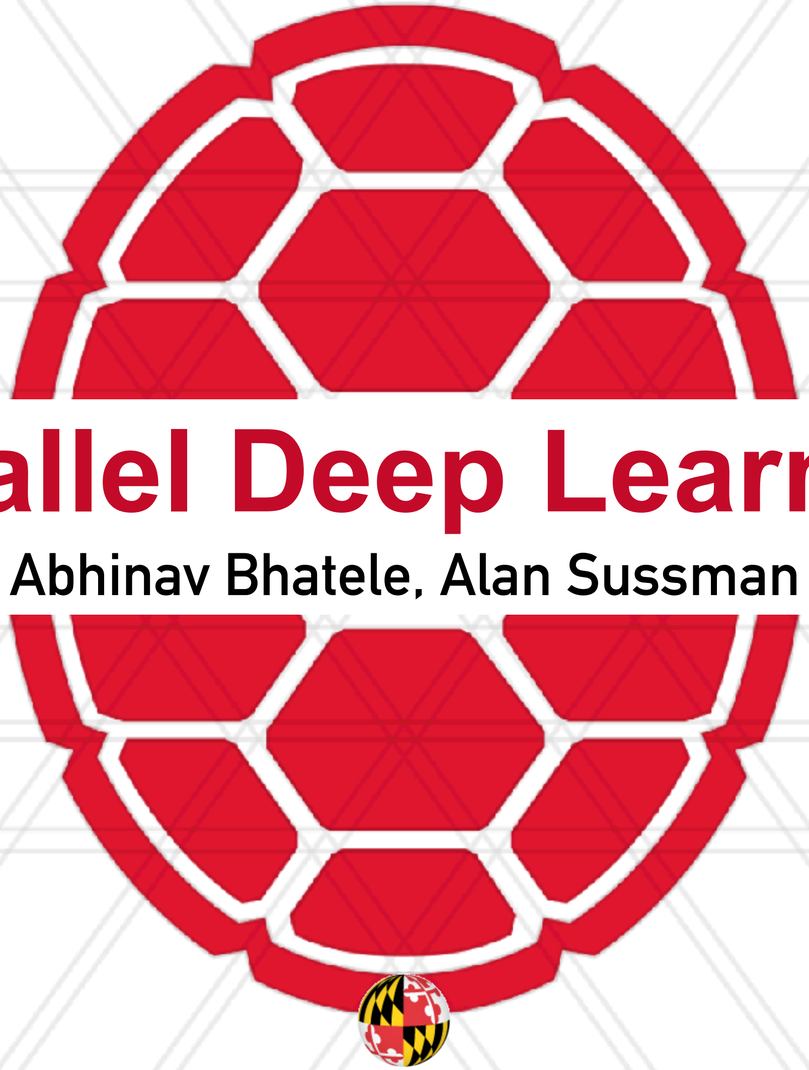
Introduction to Parallel Computing (CMSC416 / CMSC616)



Parallel Deep Learning

Abhinav Bhatele, Alan Sussman

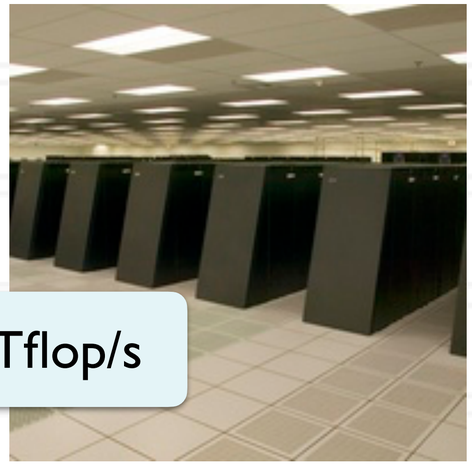
UNIVERSITY OF MARYLAND

Annoucements

- Assignment 7 (extra credit) has been posted
  - Due on Dec 9 11:59 pm (no extensions for any reason)
- IRB-approved research study to analyze the generated logs
  - Please opt-in to help us with research

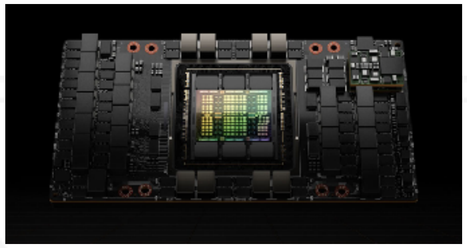
The evolution of HPC systems and rise of a new revolution in AI

- In the last two decades, an enormous amount of compute power has become available
- Large datasets and open source software such as PyTorch have also emerged
- Led to a frenzy in the world of AI and the effects are being felt in almost every other domain



Top500 Rpeak - 91.75 Tflop/s

IBM Blue Gene/L, 2004

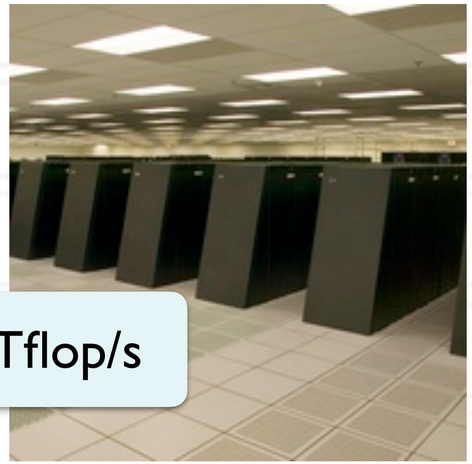


FP64 - 34 Tflop/s

NVIDIA H100, 2024

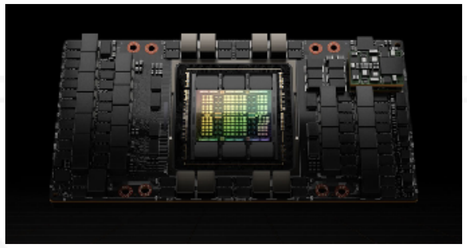
The evolution of HPC systems and rise of a new revolution in AI

- In the last two decades, an enormous amount of compute power has become available
- Large datasets and open source software such as PyTorch have also emerged
- Led to a frenzy in the world of AI and the effects are being felt in almost every other domain



Top500 Rpeak - 91.75 Tflop/s

IBM Blue Gene/L, 2004

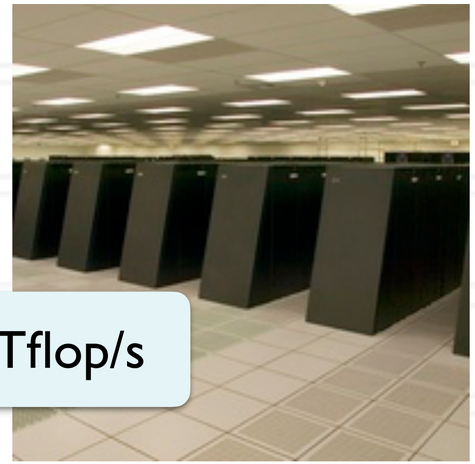


FP16 - 989 Tflop/s

NVIDIA H100, 2024

The evolution of HPC systems and rise of a new revolution in AI

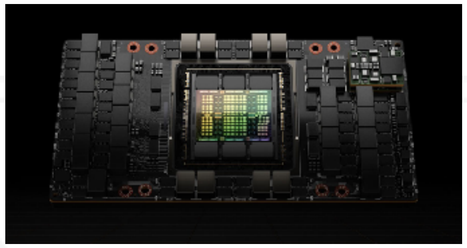
- In the last two decades, an enormous amount of compute power has become available
- Large datasets and open source software such as PyTorch have also emerged
- Led to a frenzy in the world of AI and the effects are being felt in almost every other domain



Top500 Rpeak - 91.75 Tflop/s

IBM Blue Gene/L, 2004

10.63 Exaflop/s!!



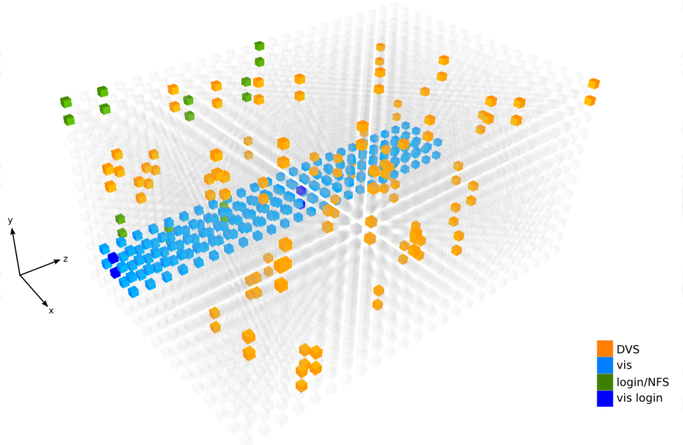
NVIDIA H100, 2024

Use of AI/ML in HPC

- Some of the earliest works were around 2007 on using traditional machine learning approaches to model performance and power
- 2007-2017: Supervised and unsupervised algorithms for creating prediction models
- 2018-present: Deep learning and transfer learning starting to become popular

Use of AI/ML in HPC

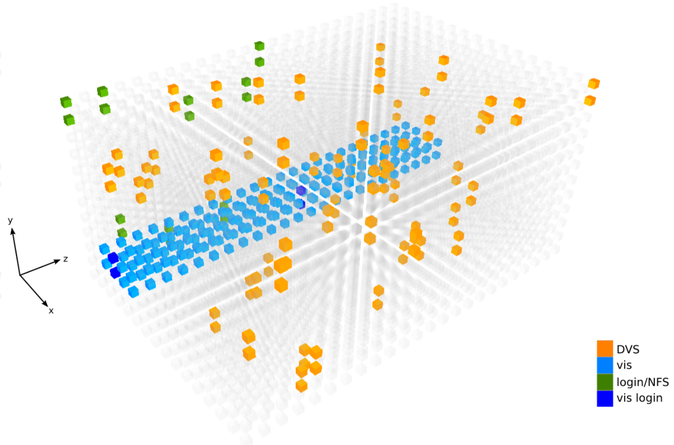
- Some of the earliest works were around 2007 on using traditional machine learning approaches to model performance and power
- 2007-2017: Supervised and unsupervised algorithms for creating prediction models
- 2018-present: Deep learning and transfer learning starting to become popular

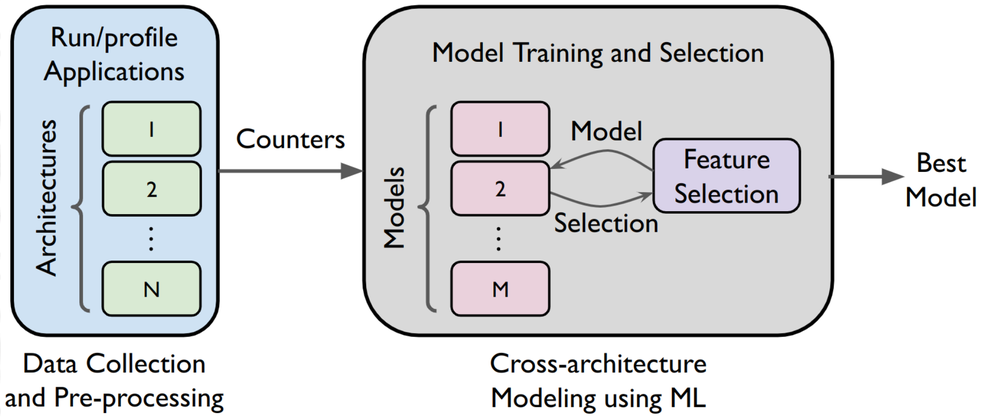


Abhinav Bhatele, et al. Identifying the culprits behind network congestion. In Proceedings of the IEEE International Parallel & Distributed Processing Symposium, IPDPS '15. May 2015.

Use of AI/ML in HPC

- Some of the earliest works were around 2007 on using traditional machine learning approaches to model performance and power
- 2007-2017: Supervised and unsupervised algorithms for creating prediction models
- 2018-present: Deep learning and transfer learning starting to become popular





Abhinav Bhatele, et al. Identifying the culprits behind network congestion. In Proceedings of the IEEE International Parallel & Distributed Processing Symposium, IPDPS '15. May 2015.

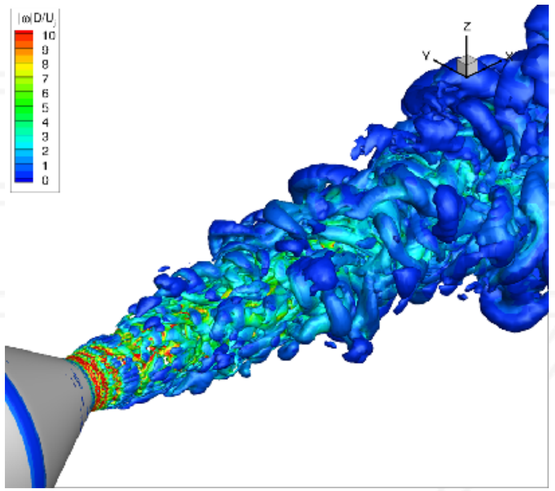
Daniel Nichols, et al. Predicting Cross-Architecture Performance of Parallel Programs. In Proceedings of the IEEE International Parallel & Distributed Processing Symposium. May 2024.

AI/ML for Computational Science

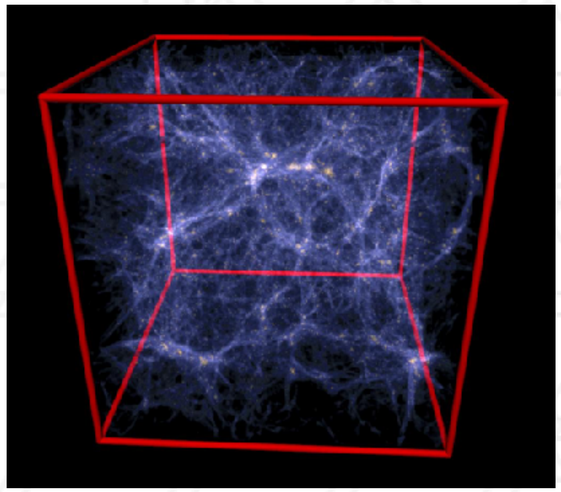
- Using of AI models for approximating computation in scientific codes
- Black box models
- Physics-informed machine learning models

AI/ML for Computational Science

- Using of AI models for approximating computation in scientific codes
- Black box models
- Physics-informed machine learning models



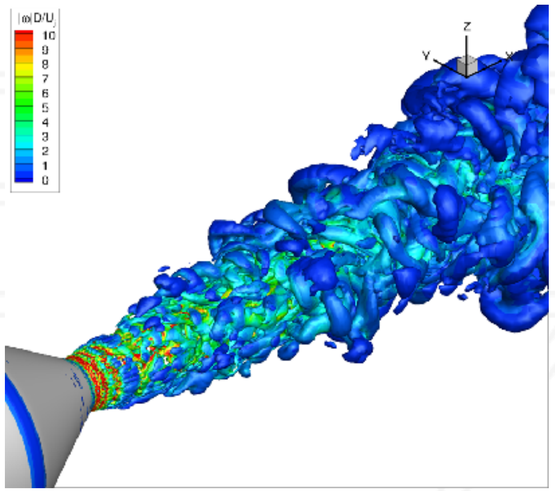
Turbulence modeling



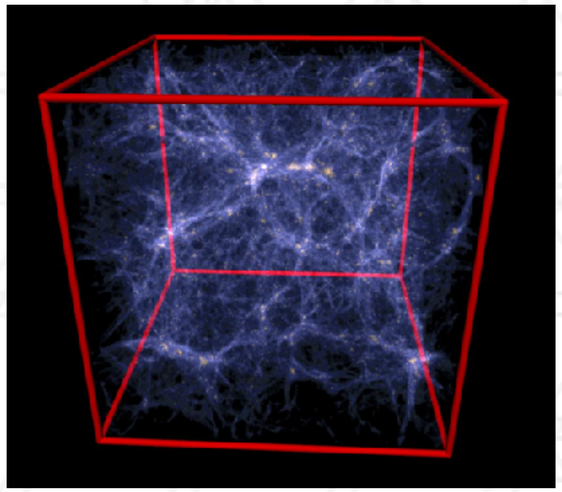
Cosmology

AI/ML for Computational Science

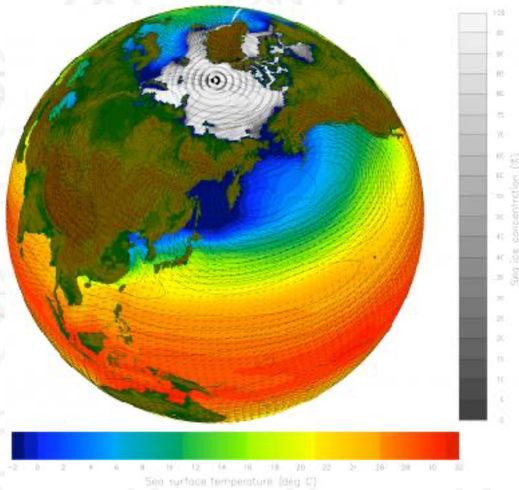
- Using of AI models for approximating computation in scientific codes
- Black box models
- Physics-informed machine learning models



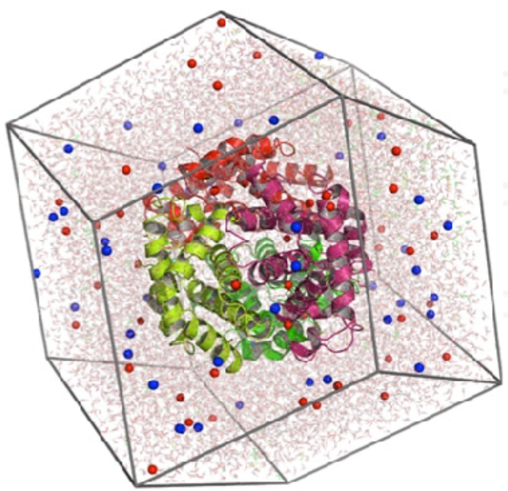
Turbulence modeling



Cosmology



Climate science



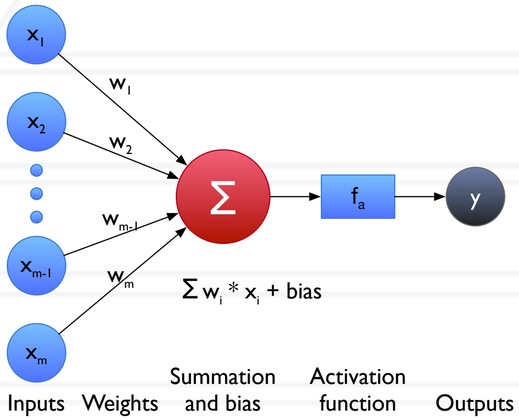
Molecular dynamics

Deep neural networks (DNNs)

- An area of machine learning that uses artificial neural networks to learn complex functions
  - Often from high-dimensional data: text, images, audio, …
- Widespread use in computer vision, natural language processing, etc.
- Neural networks can be used to model complex functions
- Several layers that process "batches" of the input data

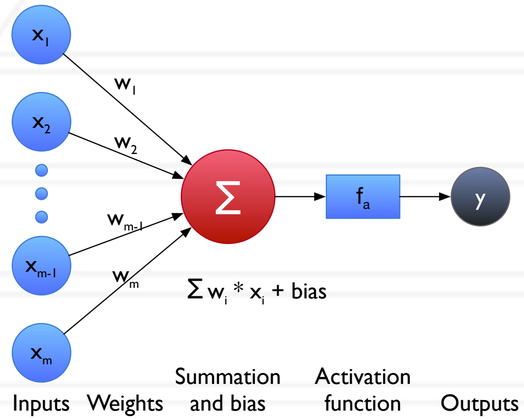
Deep neural networks (DNNs)

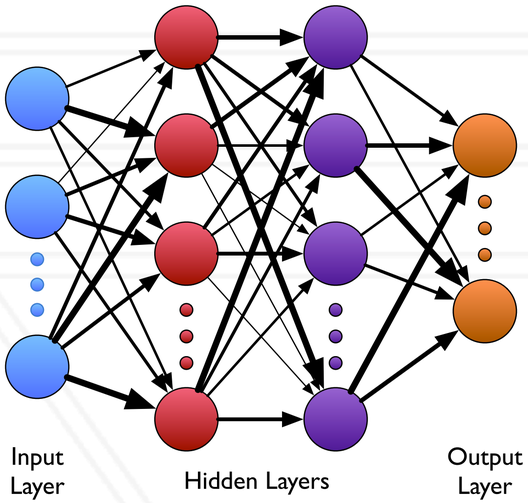
- An area of machine learning that uses artificial neural networks to learn complex functions
  - Often from high-dimensional data: text, images, audio, …
- Widespread use in computer vision, natural language processing, etc.
- Neural networks can be used to model complex functions
- Several layers that process "batches" of the input data



Deep neural networks (DNNs)

- An area of machine learning that uses artificial neural networks to learn complex functions
  - Often from high-dimensional data: text, images, audio, …
- Widespread use in computer vision, natural language processing, etc.
- Neural networks can be used to model complex functions
- Several layers that process "batches" of the input data



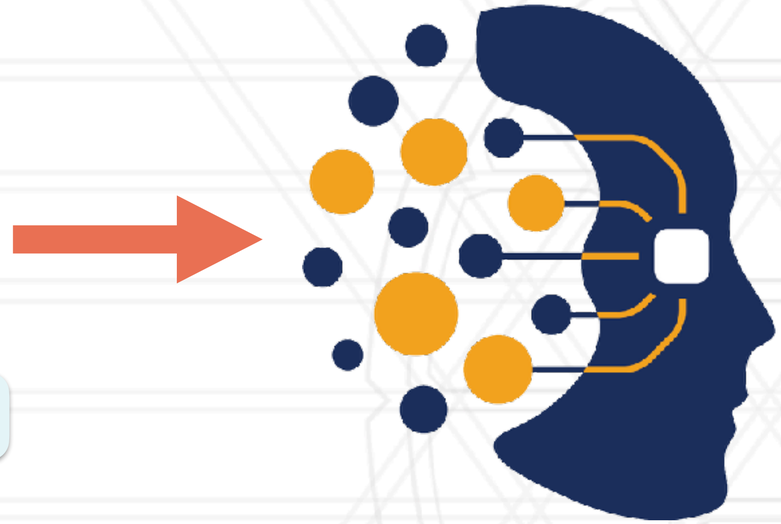


Deep learning and language models

Web data

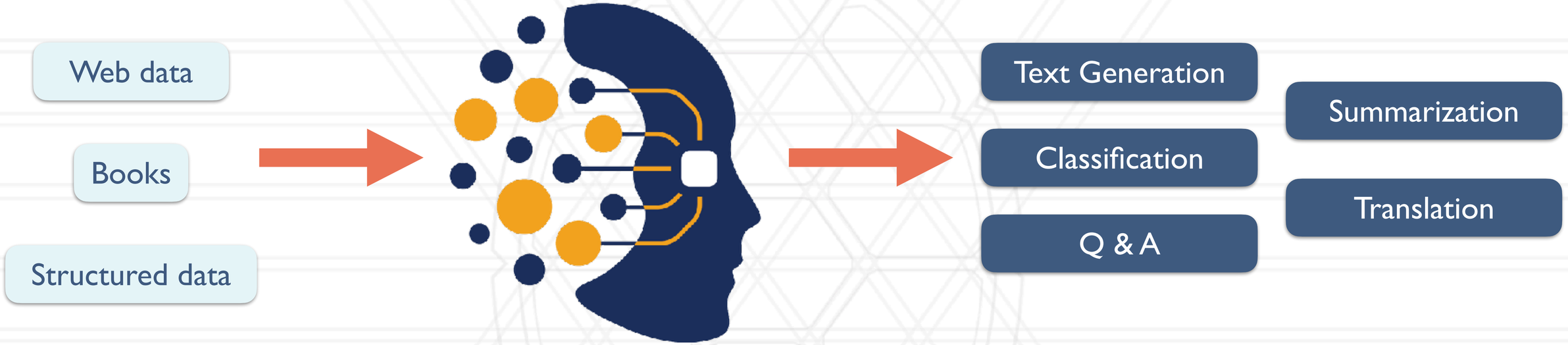
Books

Structured data



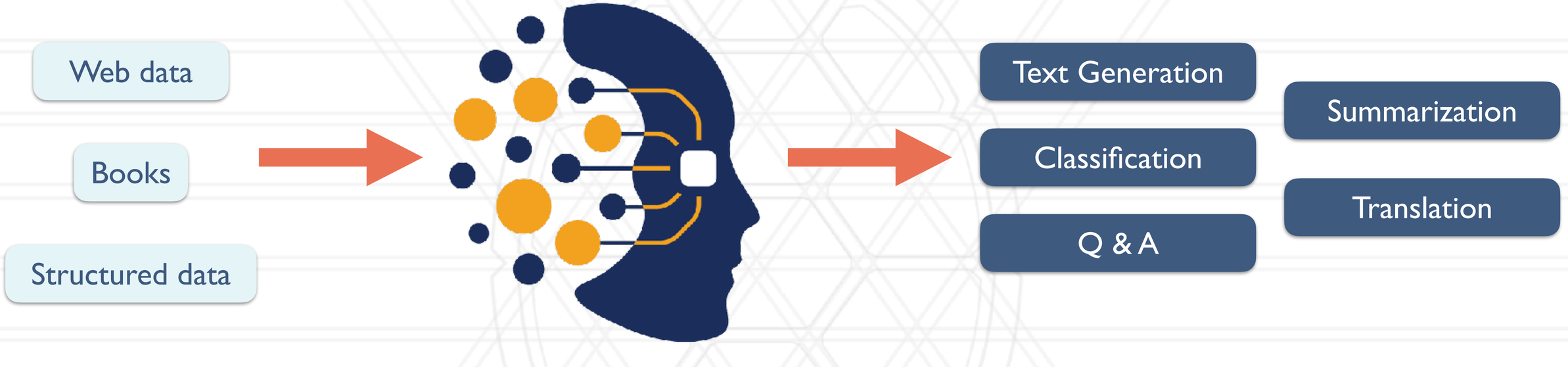
https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

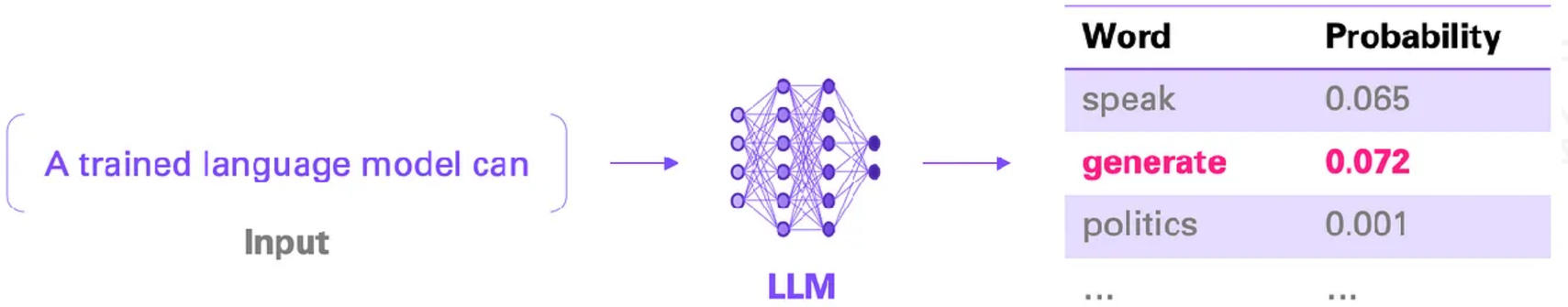
Deep learning and language models



https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

Deep learning and language models





https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

Deep learning and language models

Web data

Books

Structured data

Text Generation

Classification

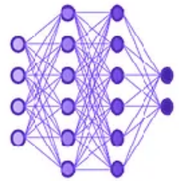
Q & A

Summarization

Translation

A trained language model can

Input



LLM

Word	Probability
speak	0.065
generate	**0.072**
politics	0.001
...	...
walk	0.003

Word	Probability
ability	0.002
text	0.084
coherent	**0.085**
...	...
ideas	0.041

https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

Other definitions

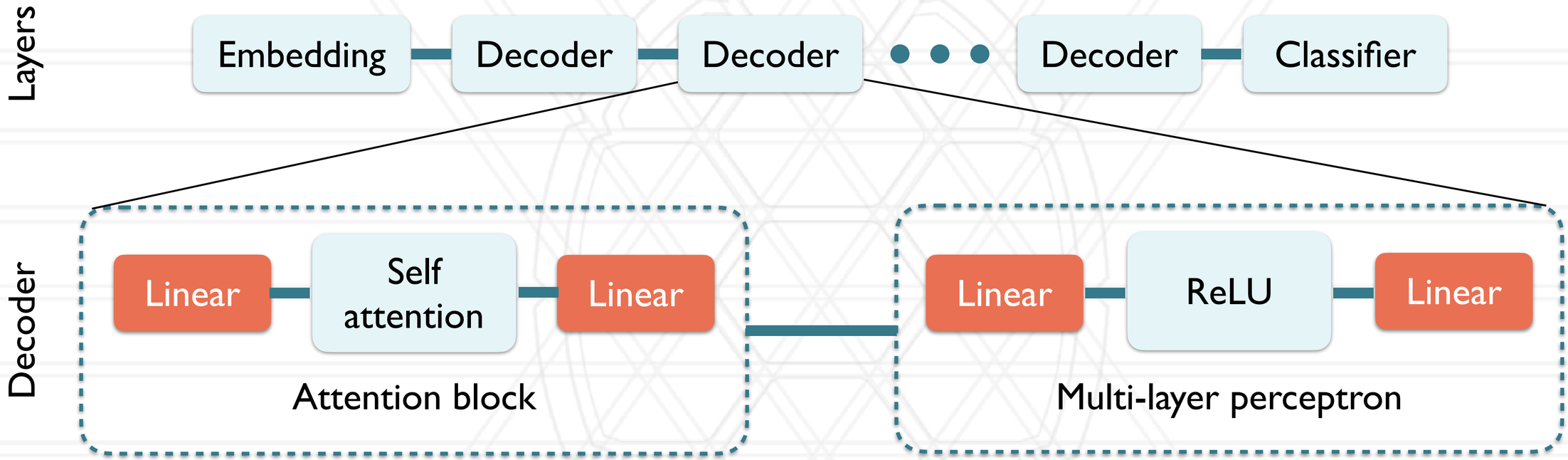
- Learning/training: task of selecting weights that lead to an accurate function
- Loss: a scalar proxy that when minimized leads to higher accuracy
- Gradient descent: process of updating the weights using gradients (derivates) of the loss weighted by a learning rate
- Mini-batch: Small subsets of the dataset processed iteratively
- Epoch: One pass over all the mini-batches

Do we really need parallel resources?

- The largest model you can run on an H100 96 GB GPU is around 3.5-4 billion parameters
- On a single node (with four H100 GPUs): around ~16 billion parameters model
- Training a 16B parameter would take 33 years!
- OpenAI's GPT 4.0 is estimated to have 1.8 trillion parameters
- Meta's Llama-3.1-405B has more than 400 billion parameters

Why is LLM training well-suited for HPC?

Why is LLM training well-suited for HPC?



Why is LLM training well-suited for HPC?

Layers

Embedding — Decoder — Decoder • • • Decoder — Classifier

Decoder

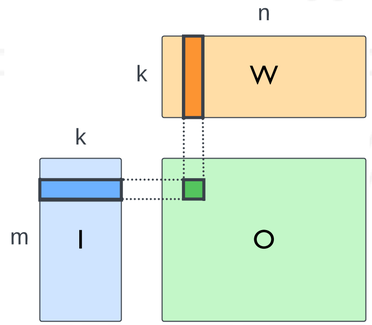
Linear — Self attention — Linear

Attention block

Linear — ReLU — Linear

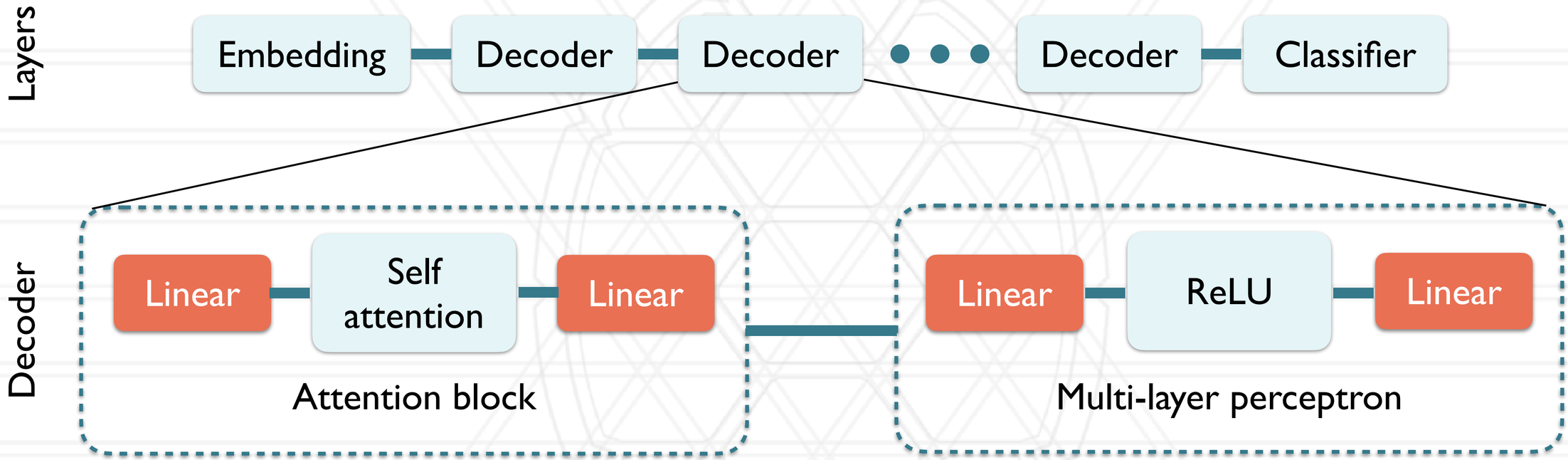
Multi-layer perceptron

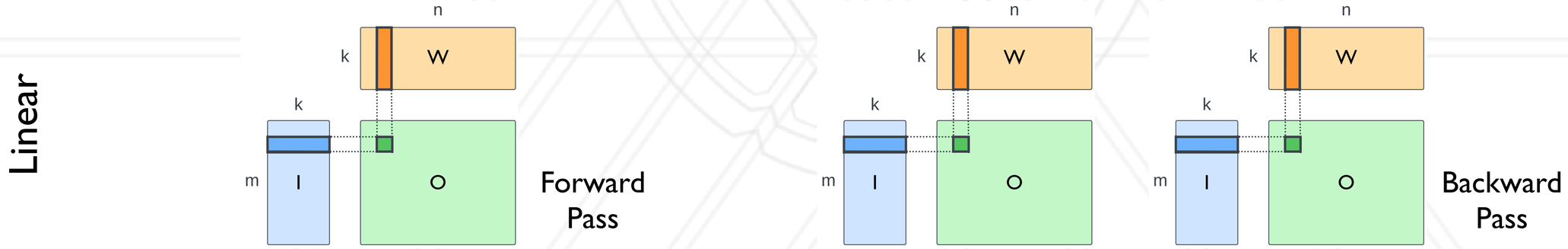
Linear



Forward Pass

Why is LLM training well-suited for HPC?



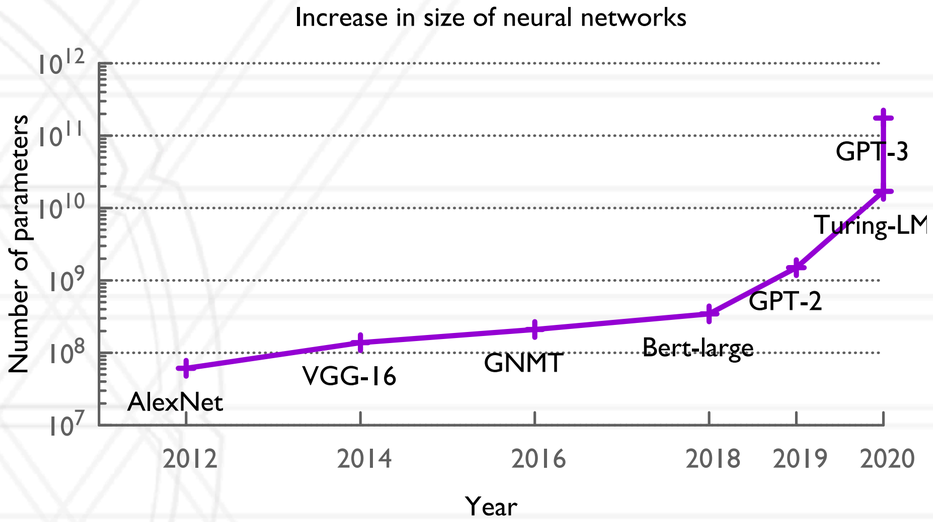


Parallel/distributed training

- Many opportunities for exploiting parallelism
- Iterative process of training (epochs)
- Many iterations per epoch (mini-batches)
- Many layers in DNNs

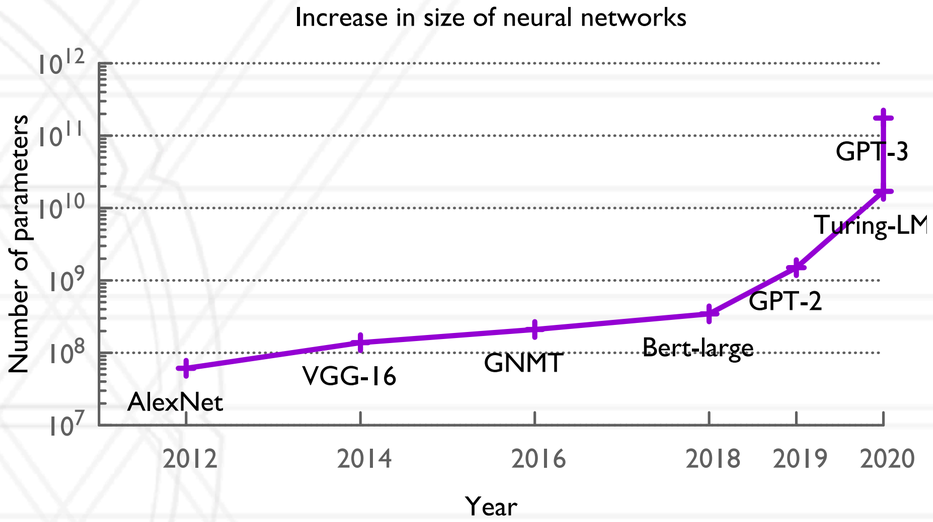
Parallel/distributed training

- Many opportunities for exploiting parallelism
- Iterative process of training (epochs)
- Many iterations per epoch (mini-batches)
- Many layers in DNNs



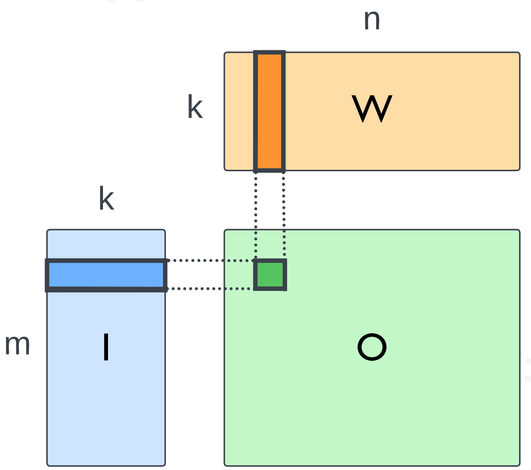
Parallel/distributed training

- Many opportunities for exploiting parallelism
- Iterative process of training (epochs)
- Many iterations per epoch (mini-batches)
- Many layers in DNNs



Framework	Type of Parallelism	Largest Accelerator Count	Largest Trained Network (No. of Parameters)
FlexFlow	Hybrid	64 GPUs	24M*
PipeDream	Inter-Layer	16 GPUs	138M
DDP	Data	256 GPUs	345M
GPipe	Inter-Layer	8 GPUs	557M
MeshTensorFlow	Intra-Layer	512-core TPUv2	4.9B
Megatron	Intra-Layer	512 GPUs	8.3B
TorchGPipe	Inter-Layer	8 GPUs	15.8B
KARMA	Data	2048 GPUs	17B
LBANN	Data	3072 CPUs	78.6B
ZeRO	Data	400 GPUs	100B

Sequential LLM training



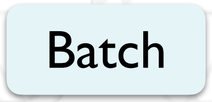

```
while (remaining_batches) {
    Read a single batch

    Forward pass: perform matrix multiplies to compute
        output activations, and a loss on the batch

    Backward pass: matrix multiplies to compute gradients of
        the loss w.r.t. parameters via backpropagation

    Optimizer step: use gradients to update the weights or
        parameters such that loss is gradually reduced
}
```

Data parallelism

- Divide training data (input batch) among workers (GPUs)
- Each worker has a full copy of the entire NN and processes different mini-batches
- All reduce operation to synchronize gradients
- Example: PyTorch's DDP, ZeRO

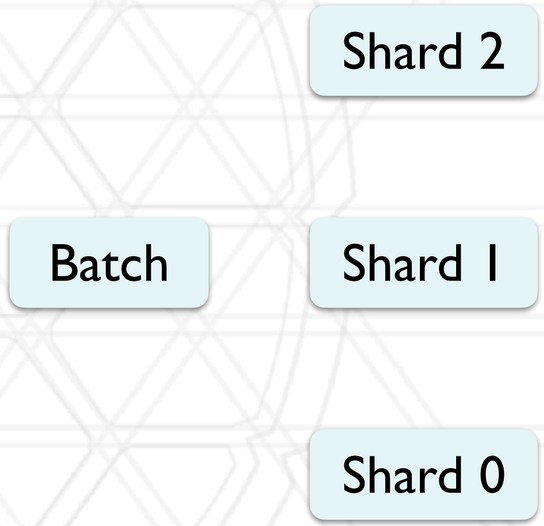
Data parallelism

- Divide training data (input batch) among workers (GPUs)
- Each worker has a full copy of the entire NN and processes different mini-batches
- All reduce operation to synchronize gradients
- Example: PyTorch's DDP, ZeRO



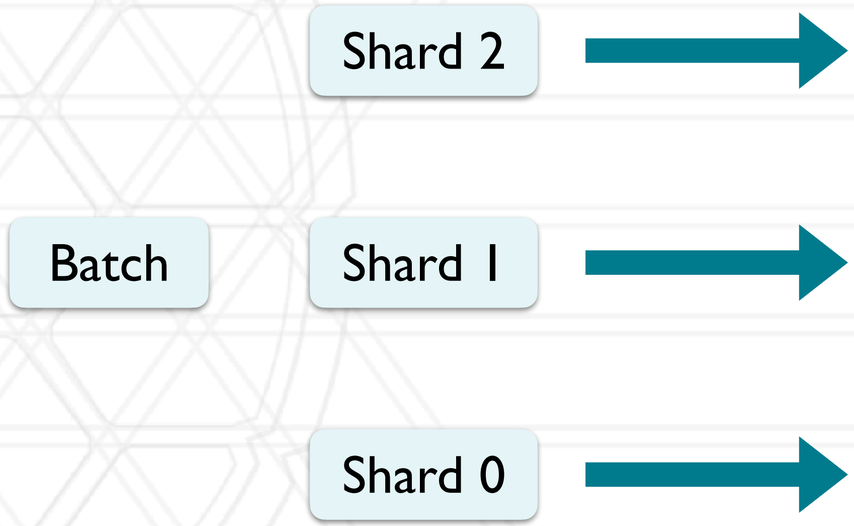
Data parallelism

- Divide training data (input batch) among workers (GPUs)
- Each worker has a full copy of the entire NN and processes different mini-batches
- All reduce operation to synchronize gradients
- Example: PyTorch's DDP, ZeRO



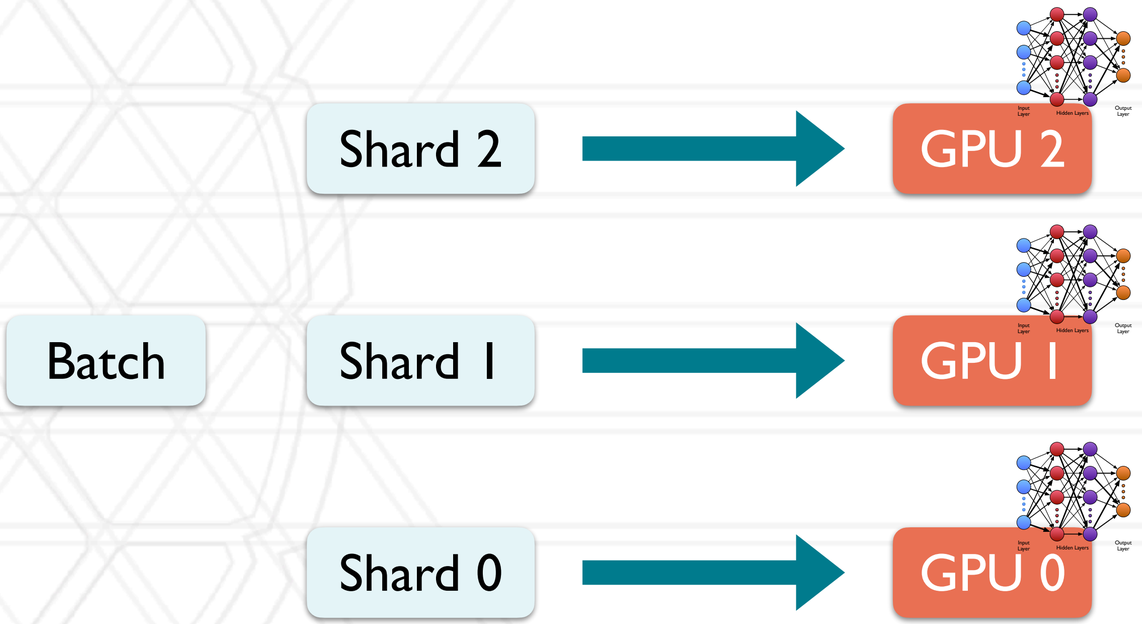
Data parallelism

- Divide training data (input batch) among workers (GPUs)
- Each worker has a full copy of the entire NN and processes different mini-batches
- All reduce operation to synchronize gradients
- Example: PyTorch's DDP, ZeRO



Data parallelism

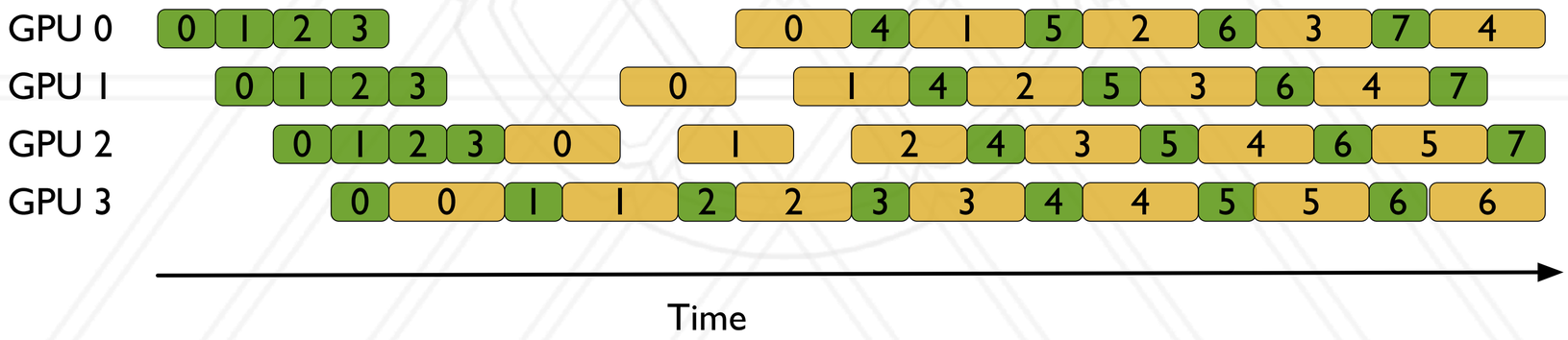
- Divide training data (input batch) among workers (GPUs)
- Each worker has a full copy of the entire NN and processes different mini-batches
- All reduce operation to synchronize gradients
- Example: PyTorch's DDP, ZeRO



Data Parallelism

Inter-layer parallelism

- Assign entire layers to different processes/GPUs
  - Ideally map contiguous subsets of layers
- Point-to-point communication (activations and gradients) between processes/GPUs managing different layers
- Use a pipeline of mini-batches to enable concurrent execution

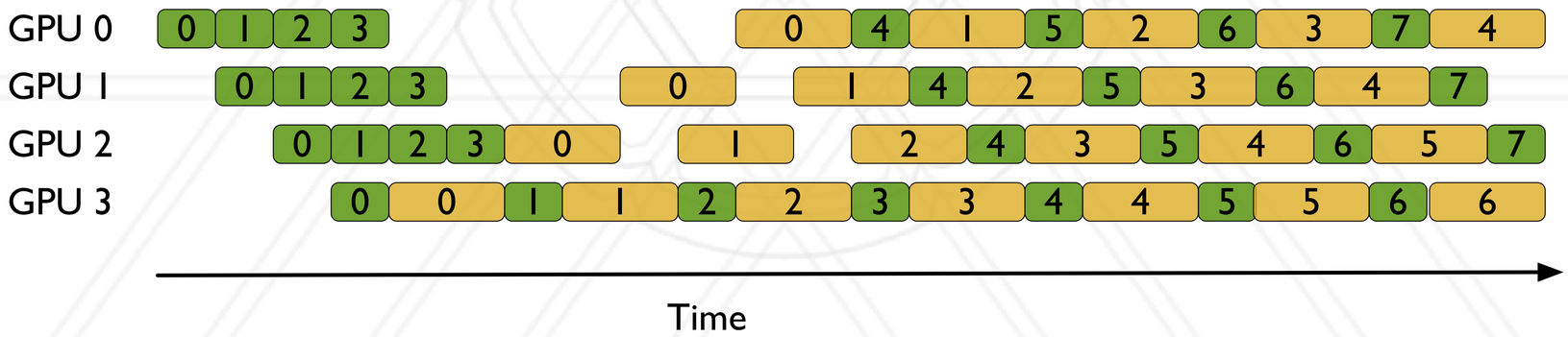


Inter-layer parallelism

- Assign entire layers to different processes/GPUs
  - Ideally map contiguous subsets of layers

Pipeline parallelism

- Point-to-point communication (activations and gradients) between processes/GPUs managing different layers
- Use a pipeline of mini-batches to enable concurrent execution



Intra-layer parallelism

- Enables training neural networks that would not fit on a single GPU
- Distribute the work within each layer to multiple processes/GPUs
  - Essentially parallelize matrix operations such as matmuls across multiple GPUs
- Example: Megatron-LM

Intra-layer parallelism

Tensor parallelism

- Enables training neural networks that would not fit on a single GPU
- Distribute the work within each layer to multiple processes/GPUs
  - Essentially parallelize matrix operations such as matmuls across multiple GPUs
- Example: Megatron-LM

Hybrid parallelism

- Using two or more approaches together in the same parallel framework
- 3D parallelism: use all three
- Popular serial frameworks: pytorch, tensorflow
- Popular parallel frameworks: DDP, MeshTensorFlow, Megatron-LM, ZeRO

A four-dimensional hybrid parallel approach

- A hybrid parallelism approach

- Combines data parallelism with 3-dimensional parallel matrix multiplication (PMM)

A four-dimensional hybrid parallel approach

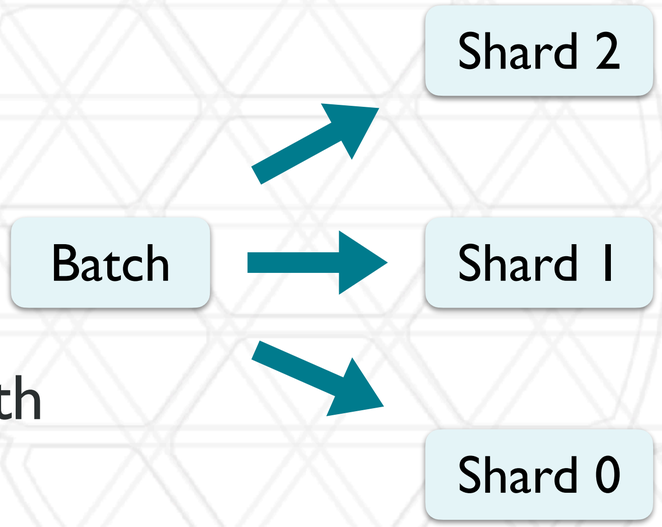
- A hybrid parallelism approach

Batch

- Combines data parallelism with 3-dimensional parallel matrix multiplication (PMM)

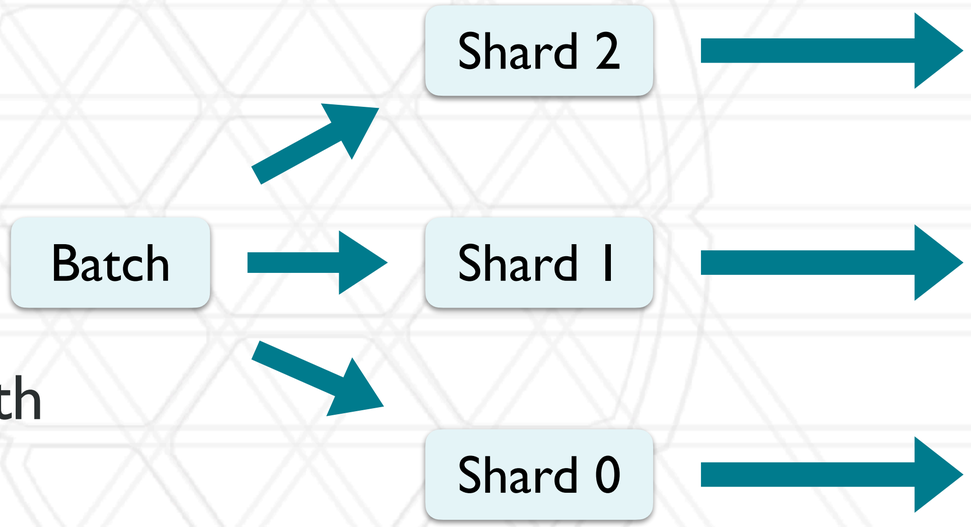
A four-dimensional hybrid parallel approach

- A hybrid parallelism approach
- Combines data parallelism with 3-dimensional parallel matrix multiplication (PMM)



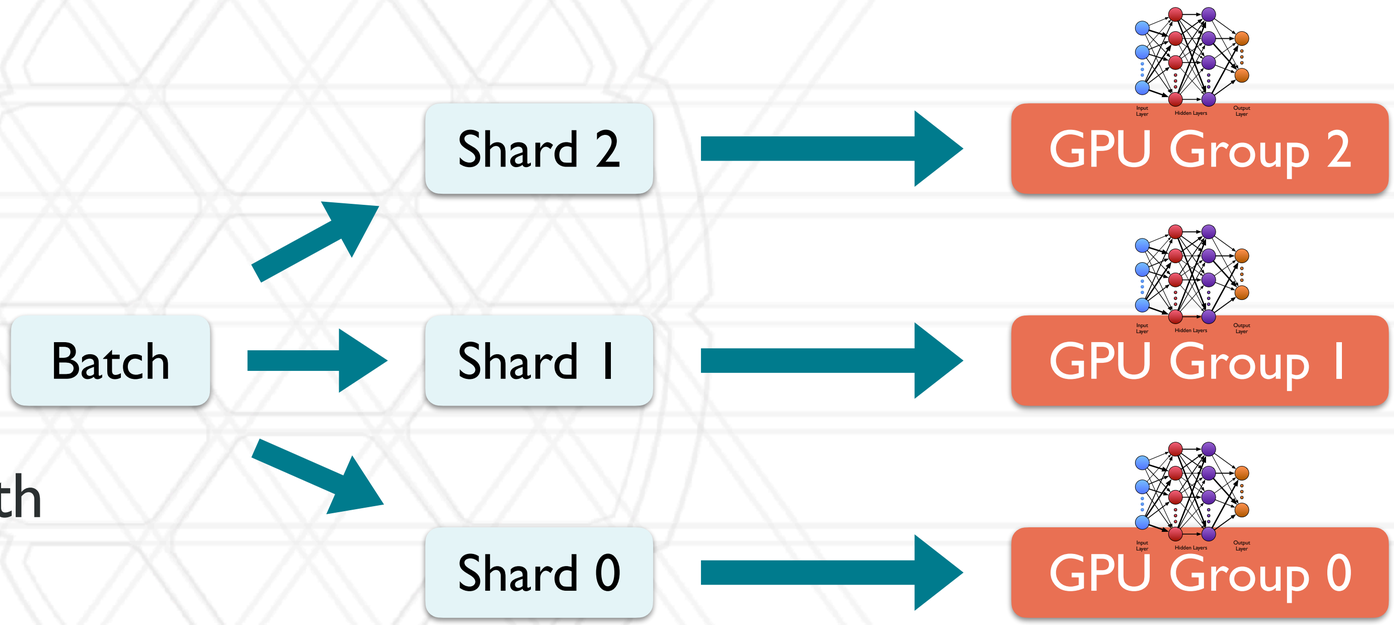
A four-dimensional hybrid parallel approach

- A hybrid parallelism approach
- Combines data parallelism with 3-dimensional parallel matrix multiplication (PMM)



A four-dimensional hybrid parallel approach

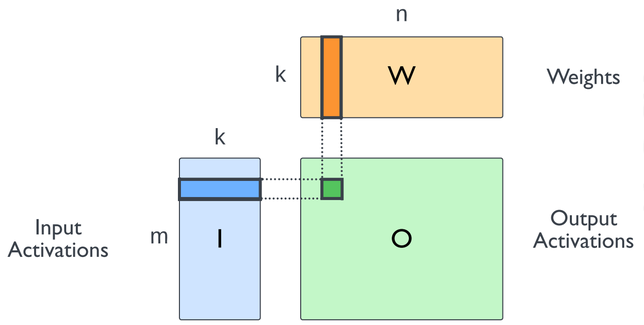
- A hybrid parallelism approach
- Combines data parallelism with 3-dimensional parallel matrix multiplication (PMM)



Data Parallelism

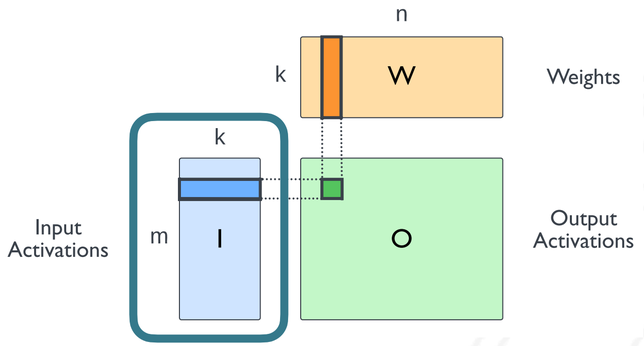
Enabling 3D parallel matrix multiplication in AxoNN

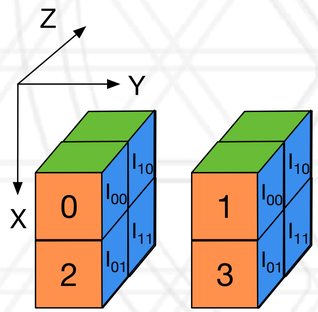
- Each layer is multiplying input activations with weights to produce output activations
- Distribute I and W across a 3D grid of GPUs
- Compute partial output activations, O on each GPU



Enabling 3D parallel matrix multiplication in AxoNN

- Each layer is multiplying input activations with weights to produce output activations
- Distribute I and W across a 3D grid of GPUs
- Compute partial output activations, O on each GPU

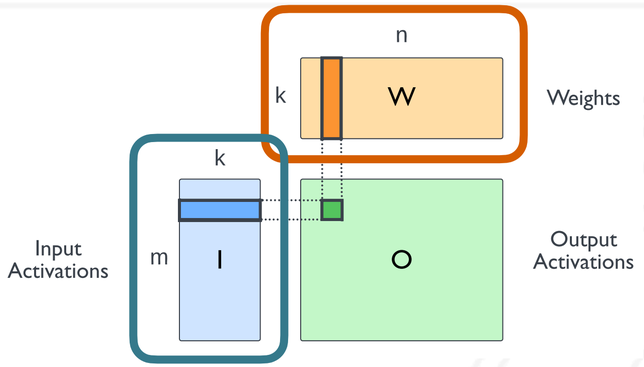


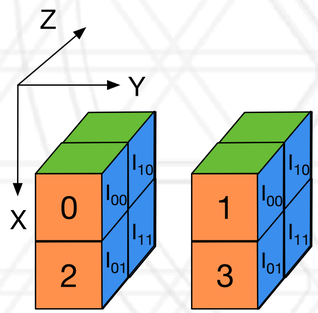


I

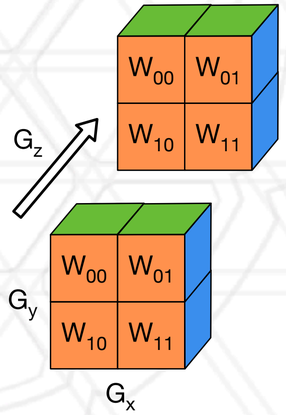
Enabling 3D parallel matrix multiplication in AxoNN

- Each layer is multiplying input activations with weights to produce output activations
- Distribute I and W across a 3D grid of GPUs
- Compute partial output activations, O on each GPU





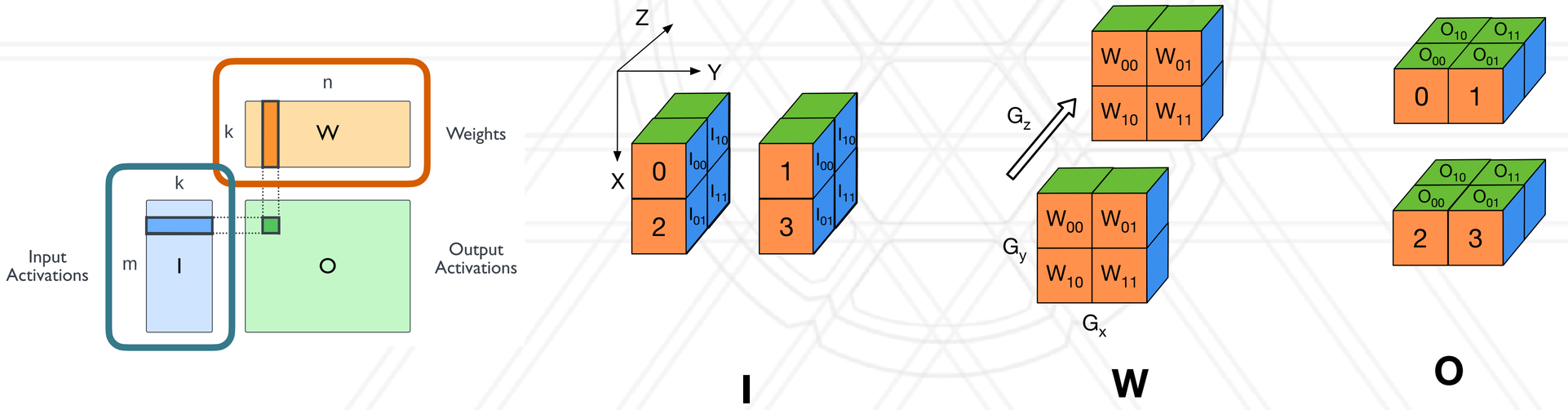
I



W

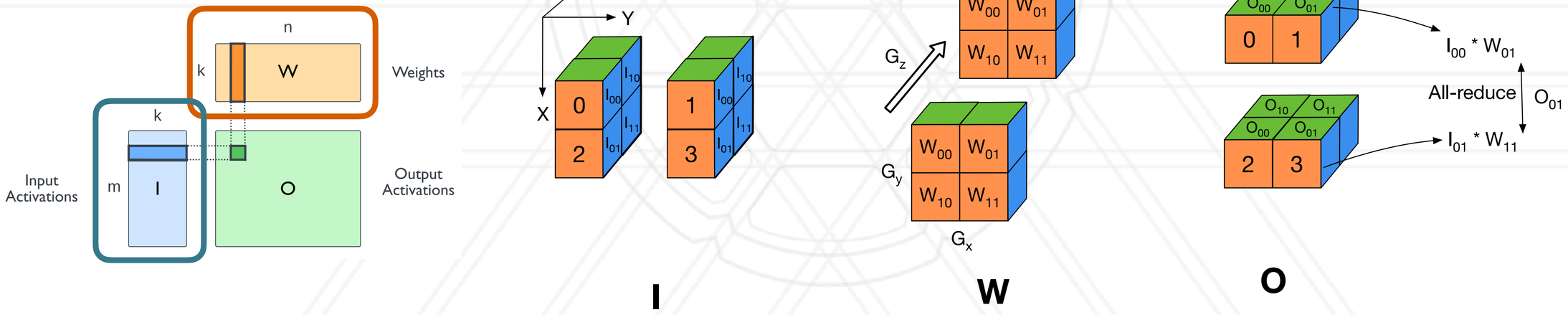
Enabling 3D parallel matrix multiplication in AxoNN

- Each layer is multiplying input activations with weights to produce output activations
- Distribute I and W across a 3D grid of GPUs
- Compute partial output activations, O on each GPU



Enabling 3D parallel matrix multiplication in AxoNN

- Each layer is multiplying input activations with weights to produce output activations
- Distribute I and W across a 3D grid of GPUs
- Compute partial output activations, O on each GPU



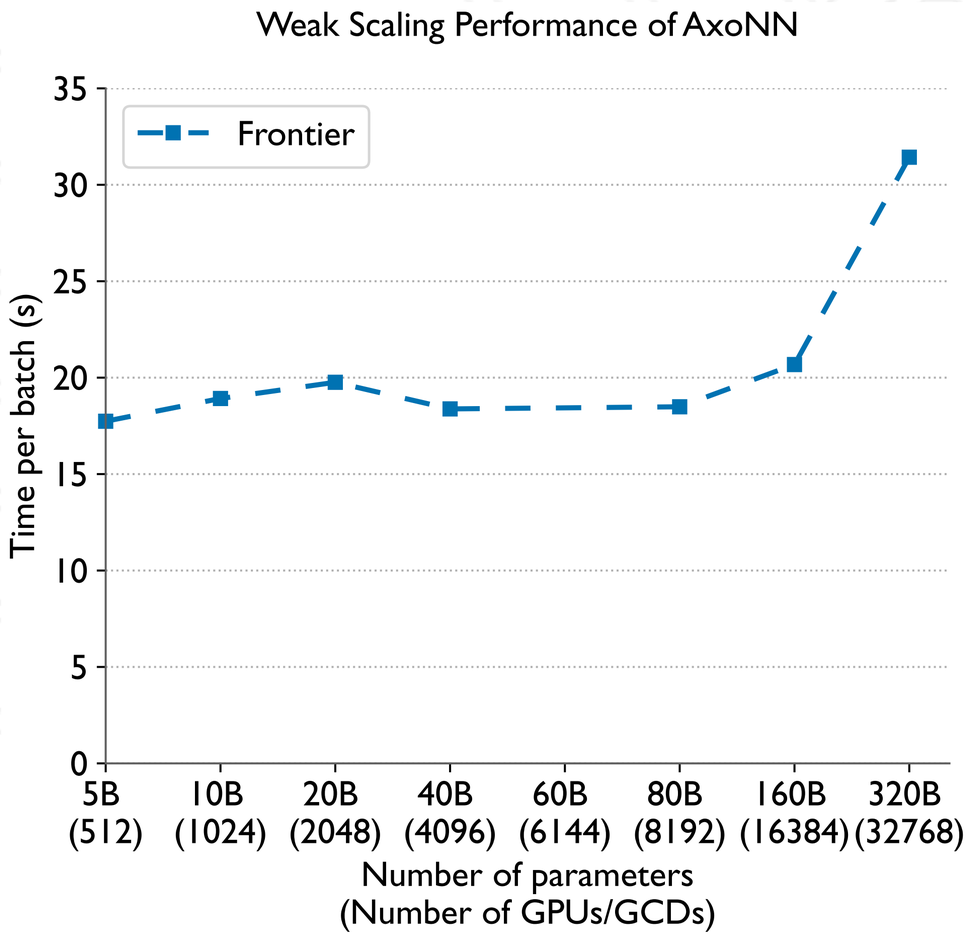
5 Easy parallelization using AxoNN

- Requires minimal code changes to model architecture (code):

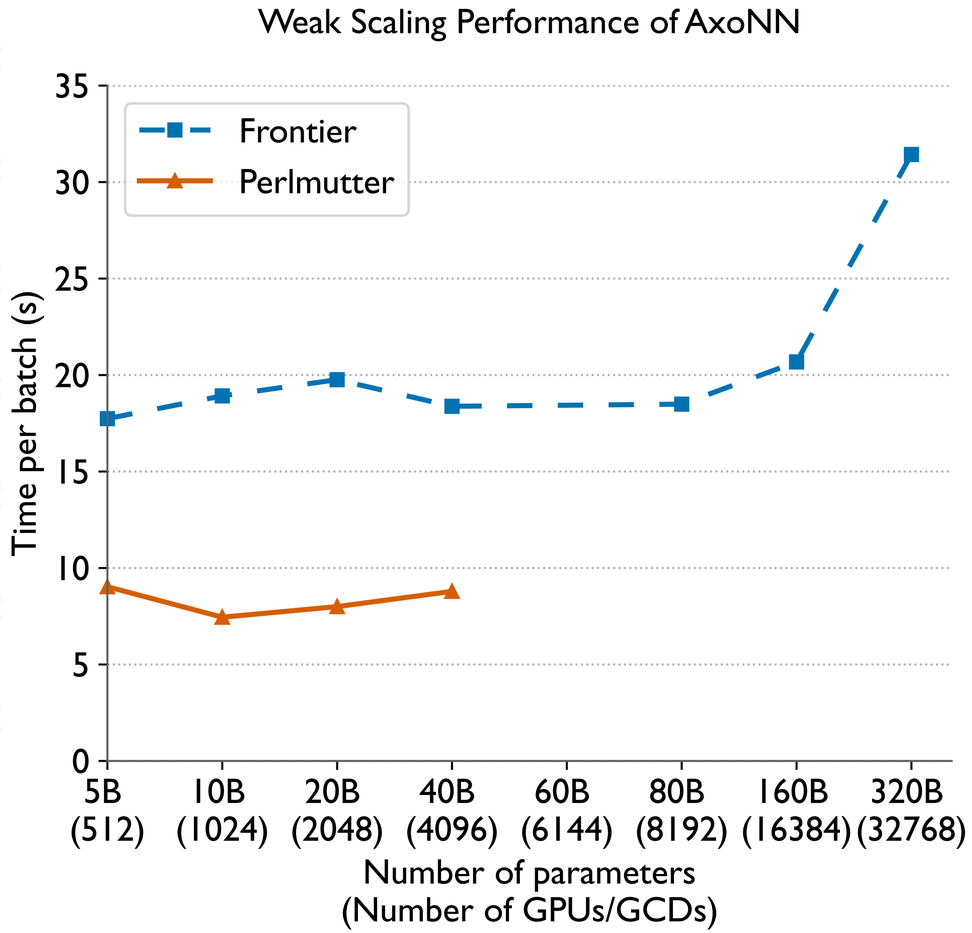
```
from axonn.intra_layer import auto_parallelize

with auto_parallelize():
    net = # declare your sequential model here
```

- AxoNN intercepts all declarations of `torch.nn.Linear`, and parallelizes them
- Our ML collaborators used this mode for the memorization experiments
- We also have backends for `lightning` and `accelerate`

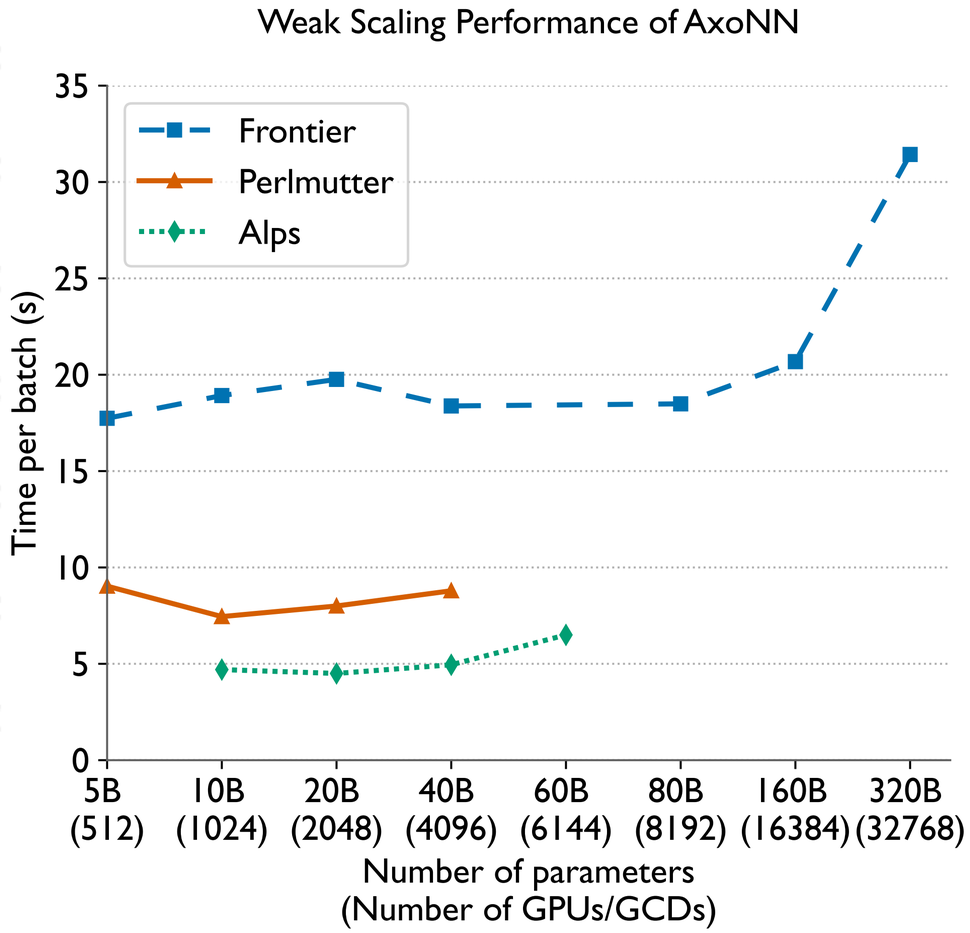
Weak scaling performance



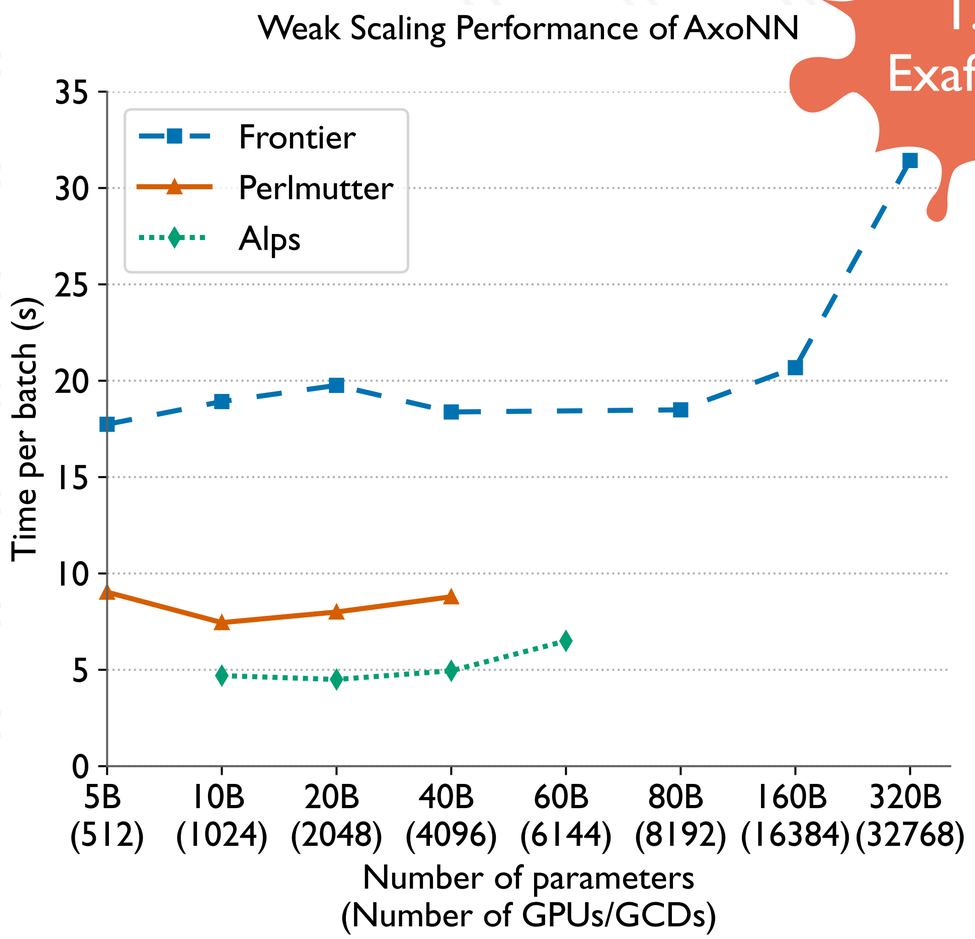
Weak scaling performance



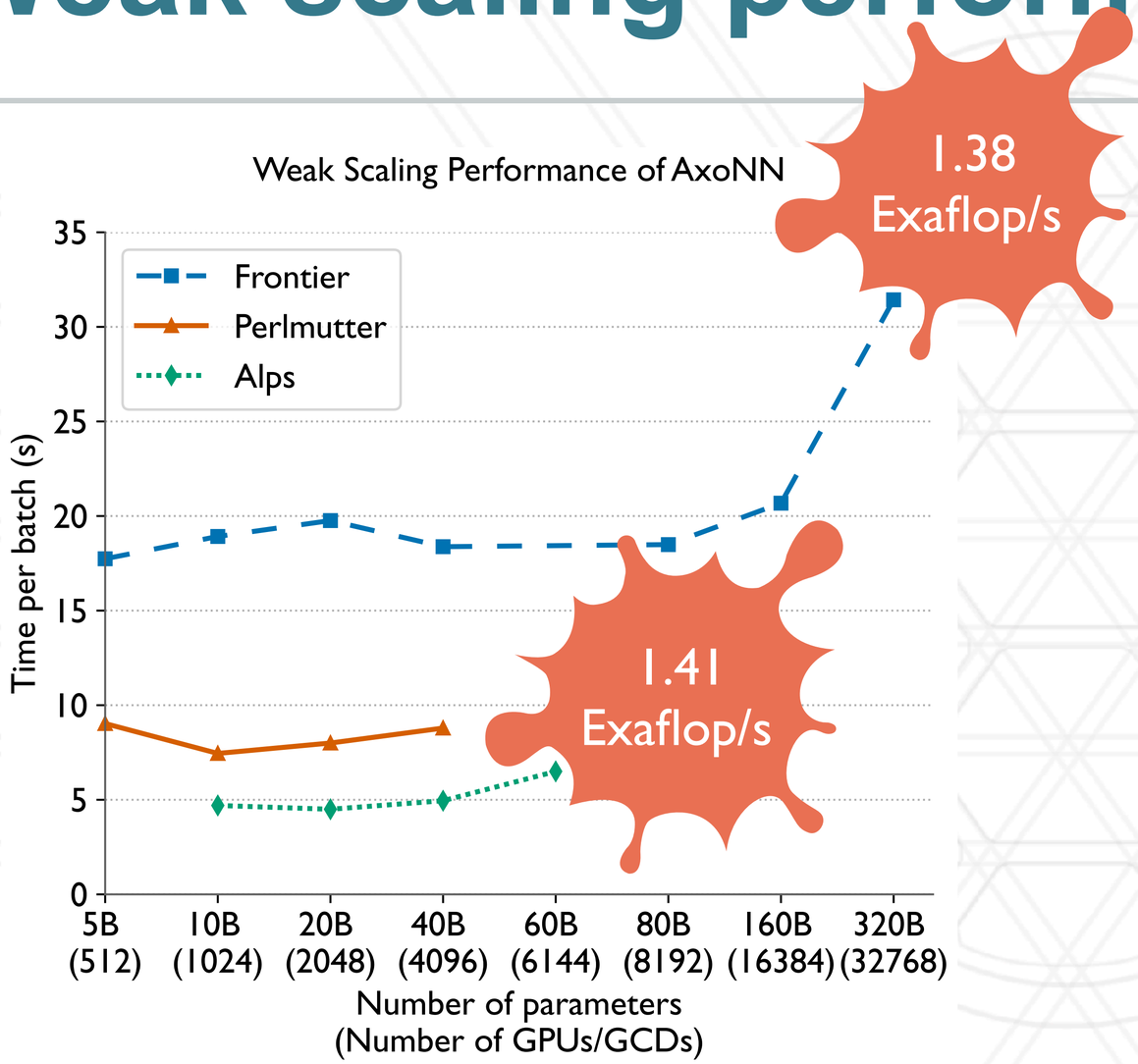
Weak scaling performance



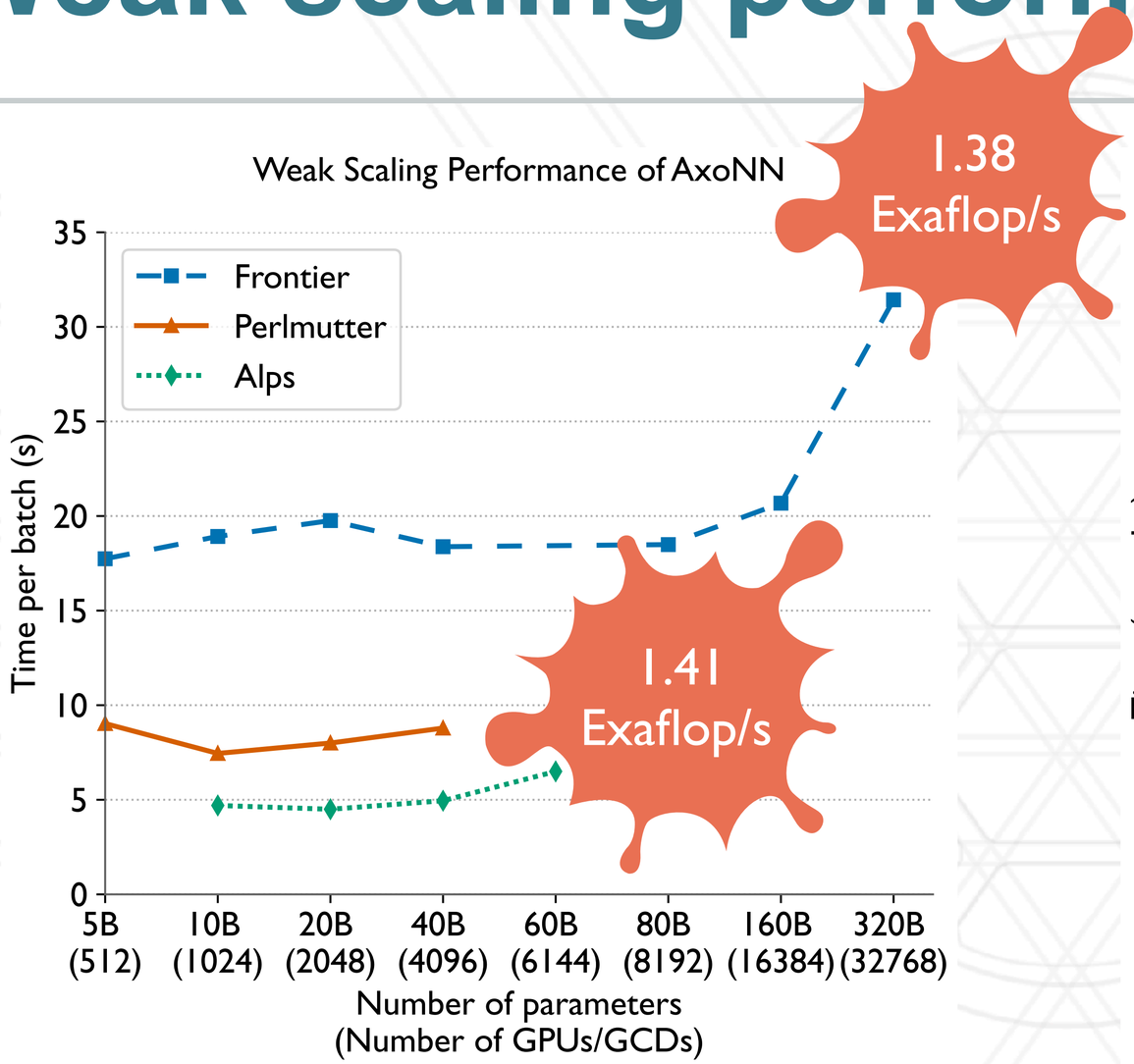
Weak scaling performance

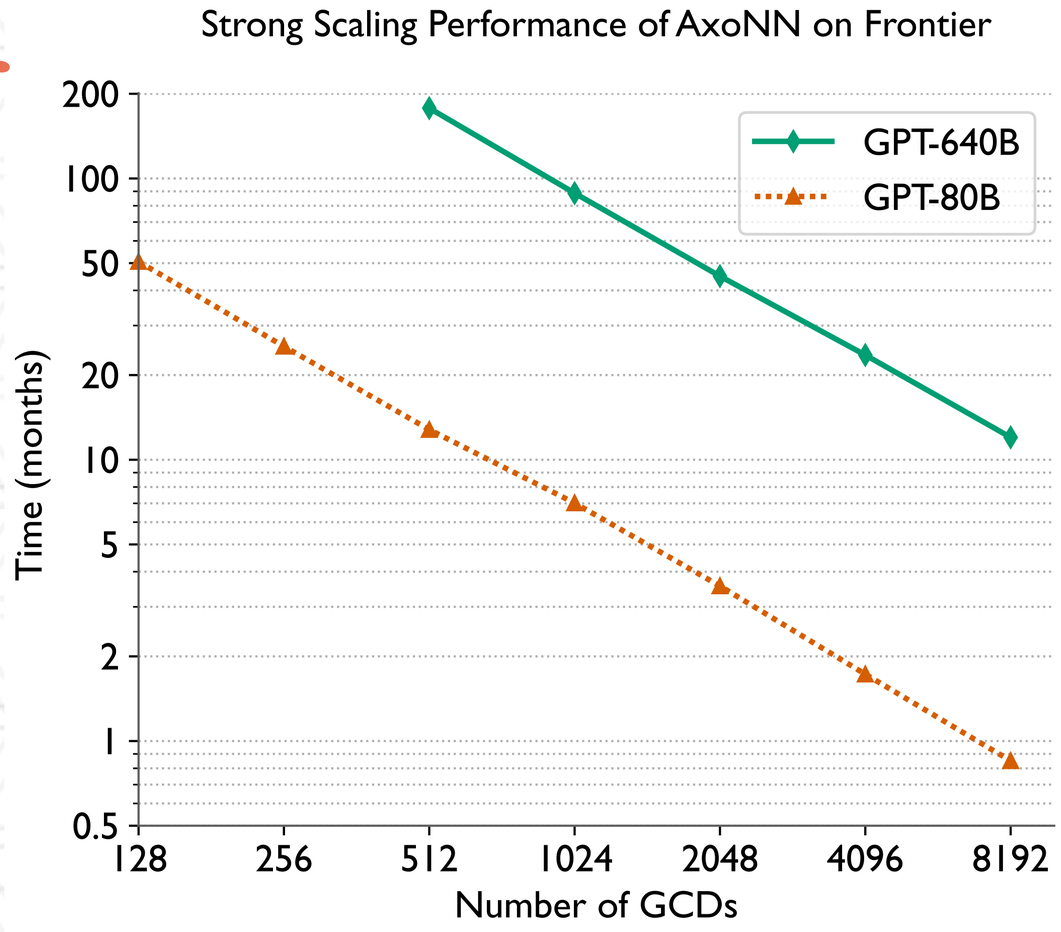


Weak scaling performance



Weak scaling performance





Questions?

UNIVERSITY OF
MARYLAND