MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Follow us on

https://twitter.com/mvapich

HiDL High-Performance Deep Learning

MPI-driven Solutions towards High-Performance Deep Learning Training and Inference on Modern Clusters

Tutorial at MUG ’25

by

Nawras Alnaasan

The Ohio State University

alnaasan.1@osu.edu

https://www.linkedin.com/in/nawras-alnaasan

Jinghan Yao

The Ohio State University

yao.877@osu.edu

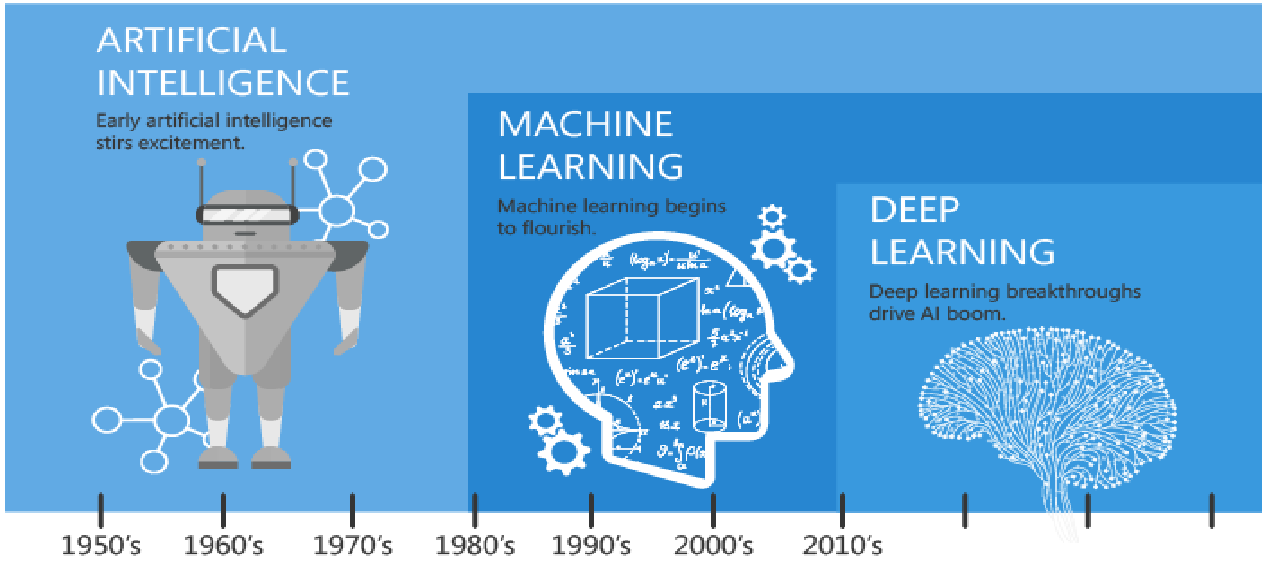
http://nowlab.cse.ohio-state.edu/member/yao.877/

Outline

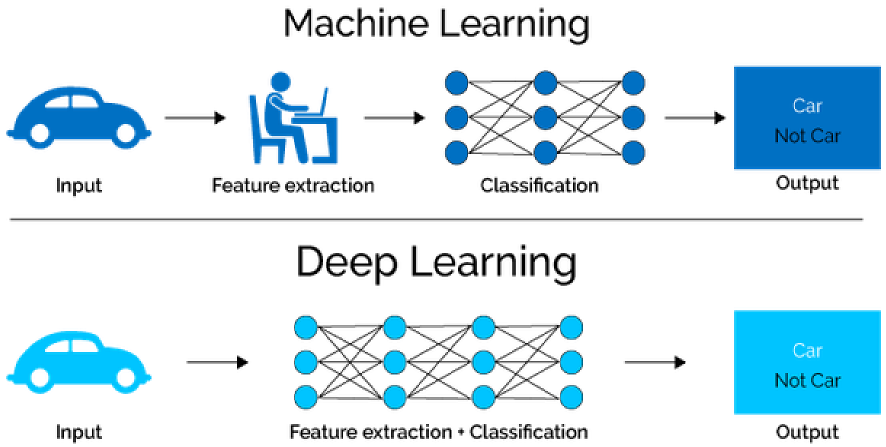
- **Introduction**
- Training
  - Deep Neural Network Training
  - Distributed Deep Learning
    - Data Parallelism
    - Sharded Data Parallelism
  - Training Solutions
- Inference
  - Tensor Parallelism
  - Expert Parallelism
- Conclusion

What is Machine Learning and Deep Learning?

- Machine Learning (ML)
  - “the study of computer algorithms to improve automatically through experience and use of data”
- Deep Learning (DL) – a subset of ML
  - Uses Deep Neural Networks (DNNs)
  - **Perhaps, the most revolutionary subset!**
- Based on learning data representation
- DNN Examples: Convolutional Neural Networks, Recurrent Neural Networks, Hybrid Networks
- Data Scientist or Developer Perspective for using DNNs
  1. Identify DL as solution to a problem
  2. Determine Data Set
  3. Select Deep Learning Algorithm to Use
  4. Use a large data set to train an algorithm

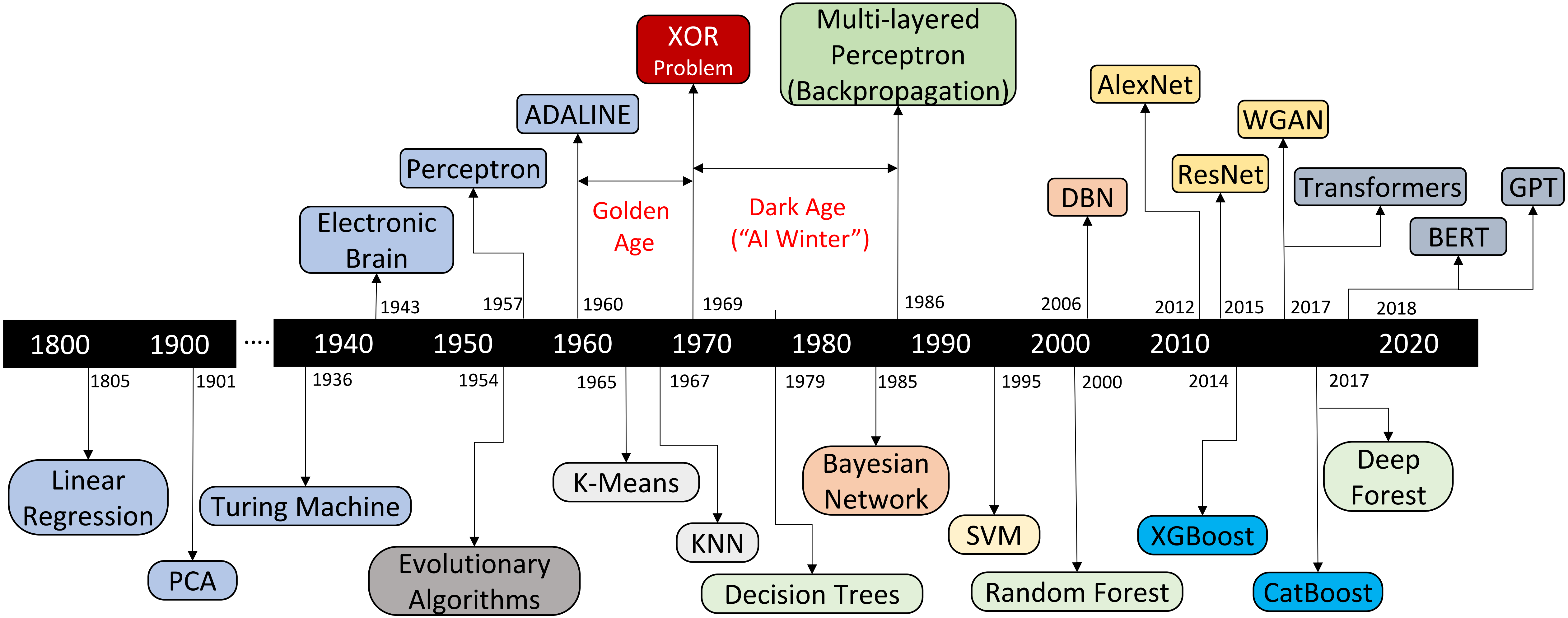


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's



Courtesy: https://hackernoon.com/difference-between-artificial-intelligence-machine-learning-and-deep-learning-1pcv3zeg, https://blog.dataiku.com/ai-vs.-machine-learning-vs.-deep-learning, https://en.wikipedia.org/wiki/Machine_learning

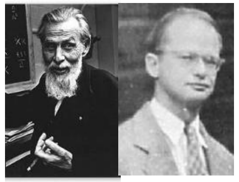
History: Milestones in the Development of ML/DL



A. Legendre – J. Gauss

K. Pearson

A. Turing


S. McCulloch – W. Pitts

F. Rosenblatt

B. Widrow – M. Hoff

M. Minsky – S. Papert

J. Pearl

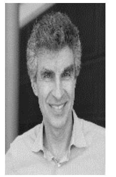
D. Rumelhart – G. Hinton – R. Wiliams

V. Vapnik– C. Cortes

A. Krizhevsky

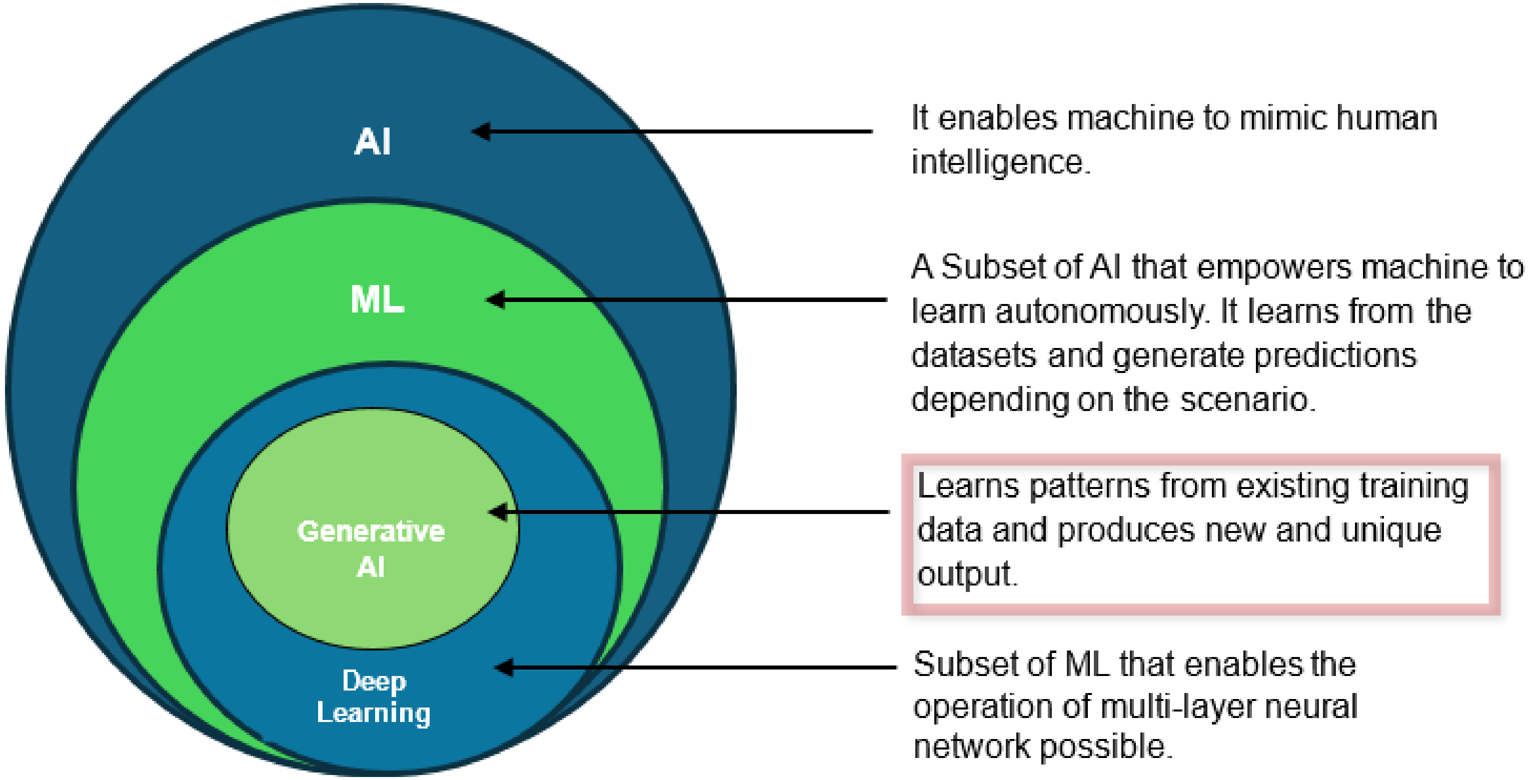
A. Ng

Y. LeCun


Y. Bengio

What is Generative AI?

- Generative AI is a subset of Deep Learning which creates new content like text, images, videos, or audio based on the data it was trained on.
- Examples:
  - Text: GPT, LLaMA, and DeepSeek.
  - Images: DALL-E and Stable Diffusion.
  - Videos: Runway and Sora.
  - Audio: AudioPaLM and VALL-E.
- What is not Generative AI?
  - Discriminative models that perform:
    - Classification
    - Regression
    - Object detection
    - Clustering
    - etc.



It enables machine to mimic human intelligence.

A Subset of AI that empowers machine to learn autonomously. It learns from the datasets and generate predictions depending on the scenario.

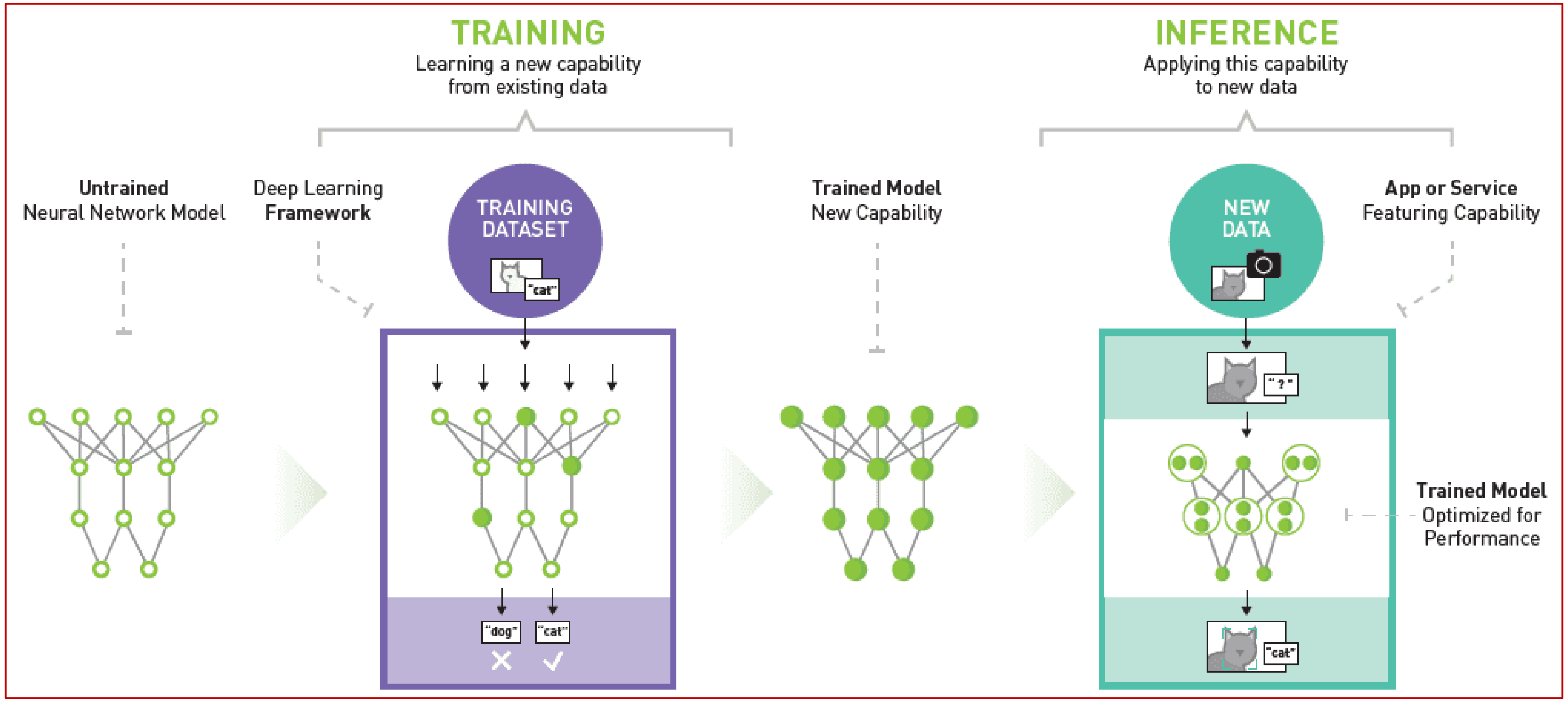
Learns patterns from existing training data and produces new and unique output.

Subset of ML that enables the operation of multi-layer neural network possible.

Courtesy: https://www.tutorialspoint.com/gen-ai/ml-and-generative-ai.htm

Training vs. Inference

- Training: the process of teaching an AI model by optimizing its parameters using training data, involving both **forward** and **backward** passes.
- Inference: the process of using a trained AI model to make predictions on new, unseen data, involving **only the forward** pass.


TRAINING
Learning a new capability from existing data
INFERENCE
Applying this capability to new data
Untrained Neural Network Model
Deep Learning Framework
TRAINING DATASET
"cat"
"dog"
"cat"
Trained Model New Capability
NEW DATA
App or Service Featuring Capability
"?"
"cat"
Trained Model Optimized for Performance

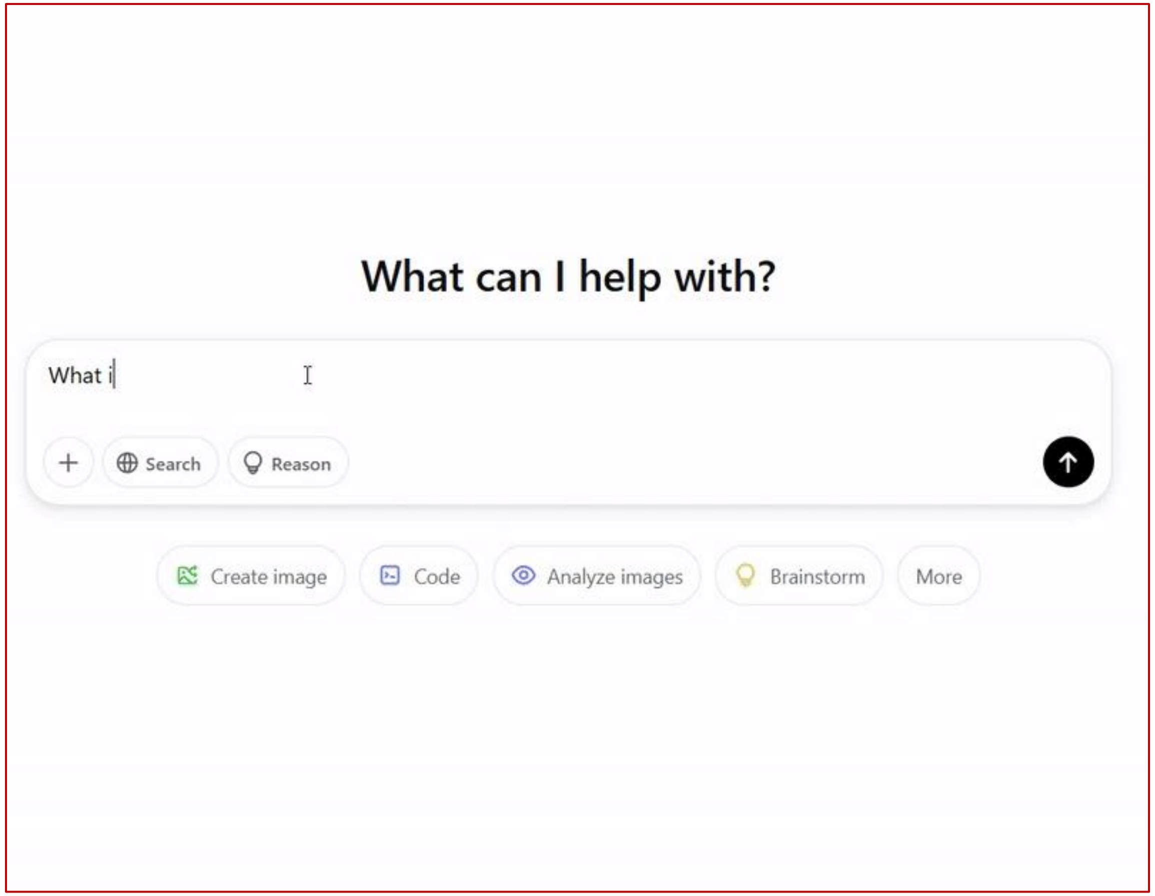
Courtesy: https://www.siemon.com/en/environments/data-centers/generative-ai/

Generative AI – Inference

In inference, the model generates outputs based on input prompts. For autoregressive models (most LLMs), inference follows an **iterative loop**, where each generated token (word) is **fed back** as input for the next step until completion.

LLM inference requires low-latency, high-throughput compute with the following key QoS (Quality of Service) requirements:

- **Low Latency** – Ensures fast response times, crucial for interactive applications.
- **Efficient Batch Processing** – Optimized for serving multiple queries in parallel to maximize throughput.
- **Mixed-Precision Support (FP16/BF16/INT8)** – Reduces compute overhead while maintaining accuracy.
- **High-Speed Interconnects (NVLink, InfiniBand)** – Required for multi-GPU inference to minimize communication bottlenecks.
- **High Memory Bandwidth** – To efficiently load large model weights and handle activation memory.



What can I help with?

What i

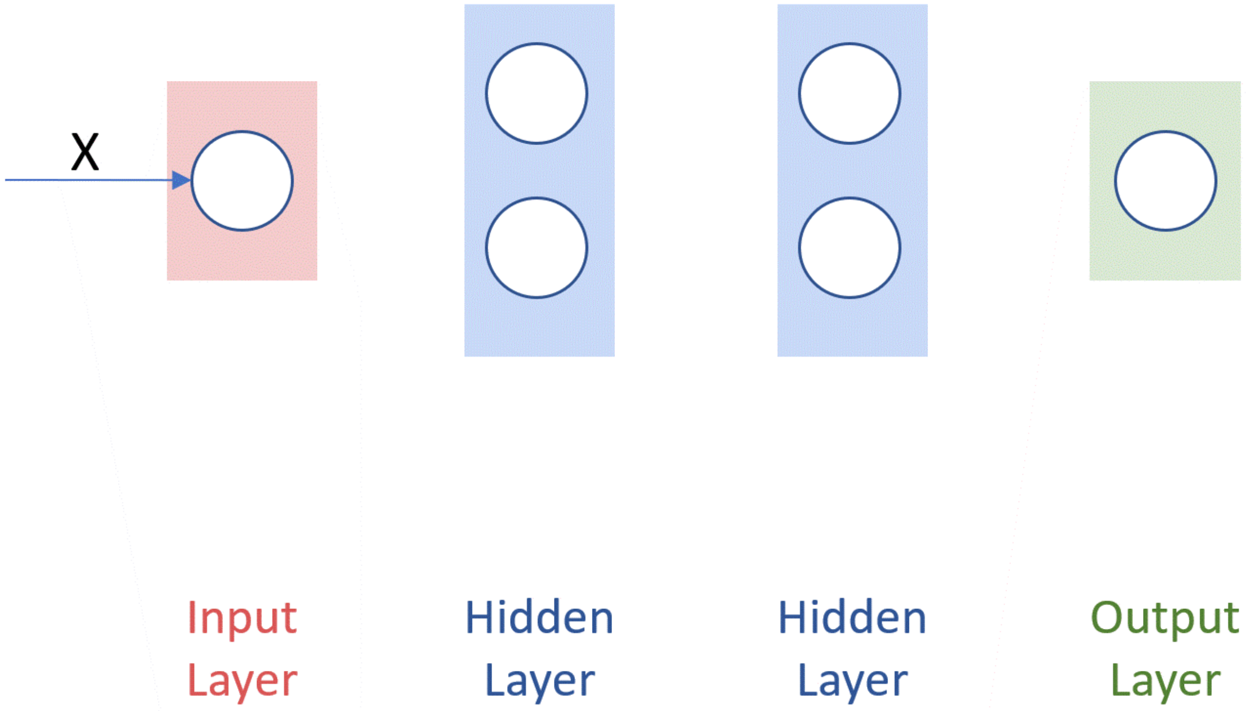
Online LLM Inferencing

Outline

- Introduction
- **Training**
  - **Deep Neural Network Training**
  - Distributed Deep Learning
    - Data Parallelism
    - Sharded Data Parallelism
  - Training Solutions
- Inference
  - Tensor Parallelism
  - Expert Parallelism
- Conclusion

Understanding the Deep Neural Network Concepts

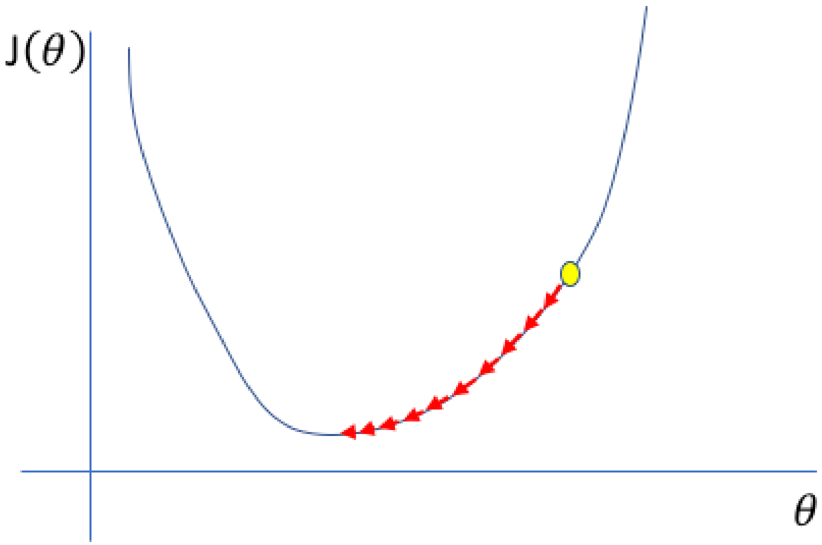
- Example of a 3-layer Deep Neural Network (DNN) – (input layer is not counted)



Courtesy: http://cs231n.github.io/neural-networks-1/

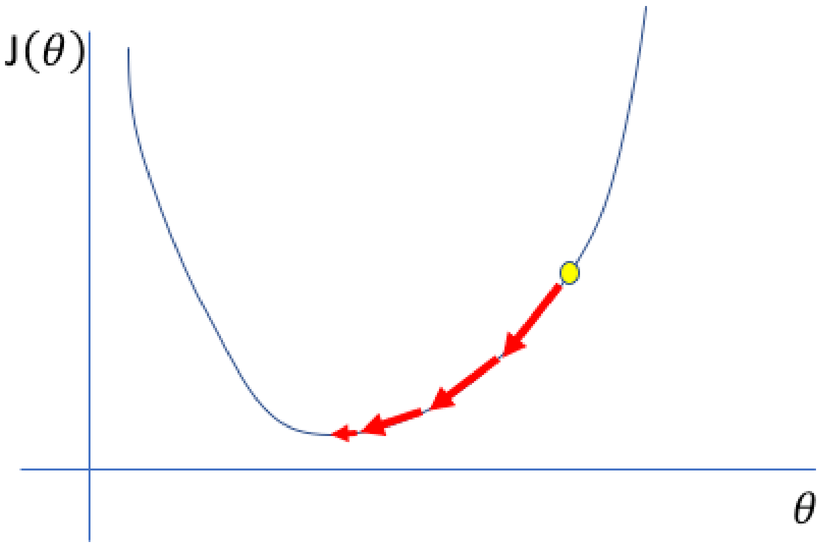
Essential Concepts: Learning Rate (α)

Too low



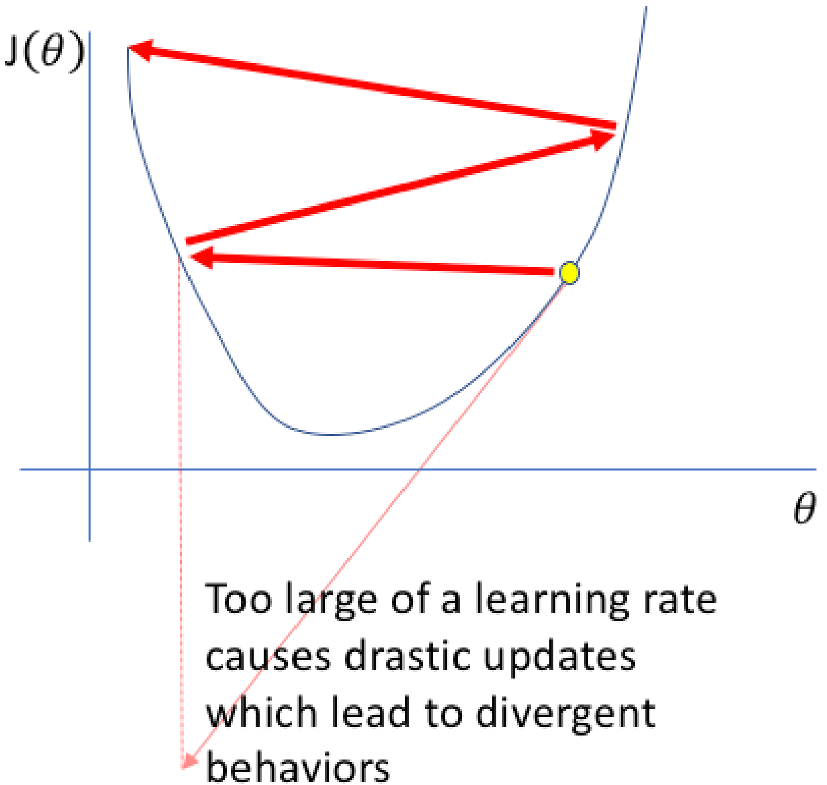
A small learning rate requires many updates before reaching the minimum point

Just right



The optimal learning rate swiftly reaches the minimum point

Too high

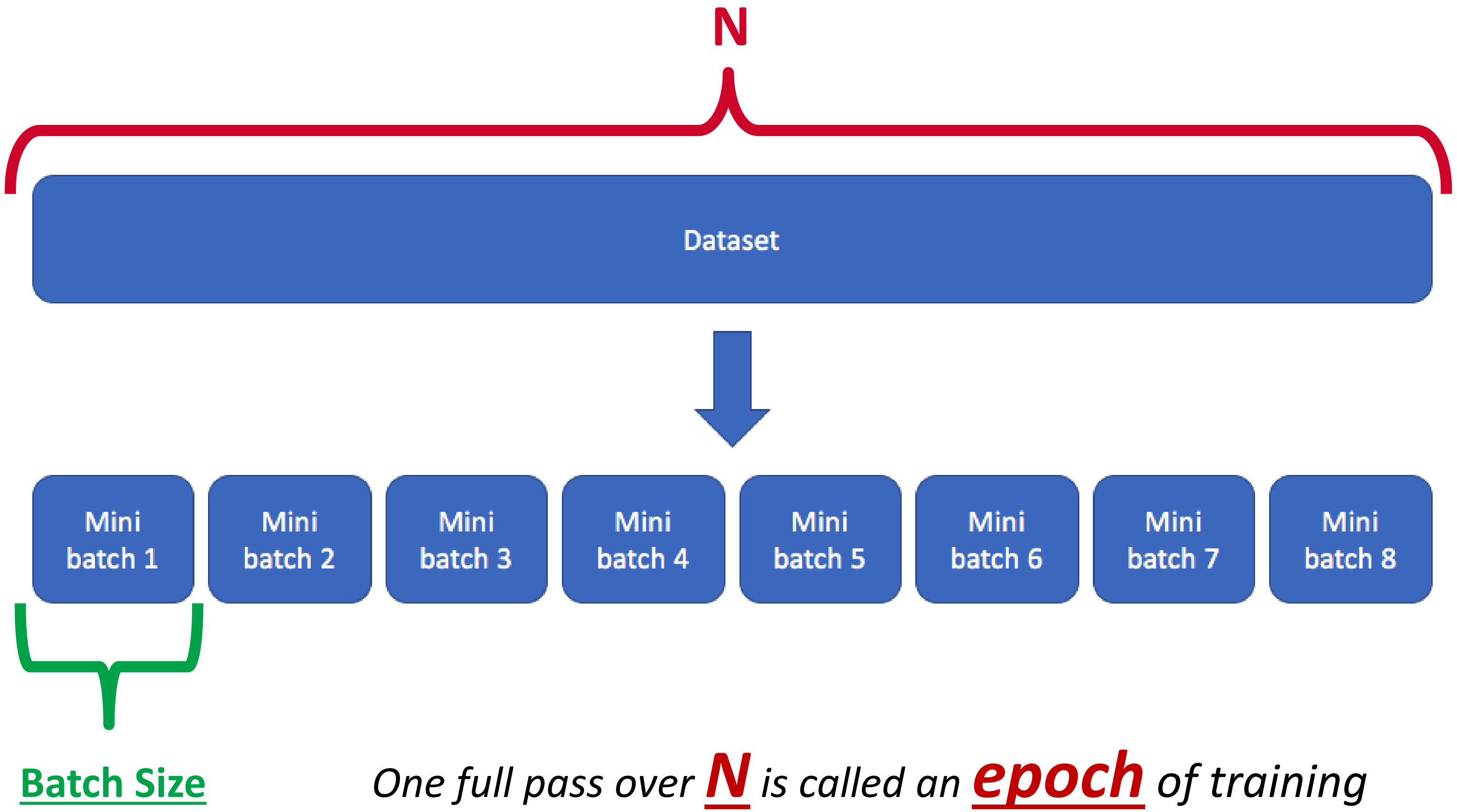


Too large of a learning rate causes drastic updates which lead to divergent behaviors

Courtesy: https://www.jeremyjordan.me/nn-learning-rate/

Essential Concepts: Batch Size

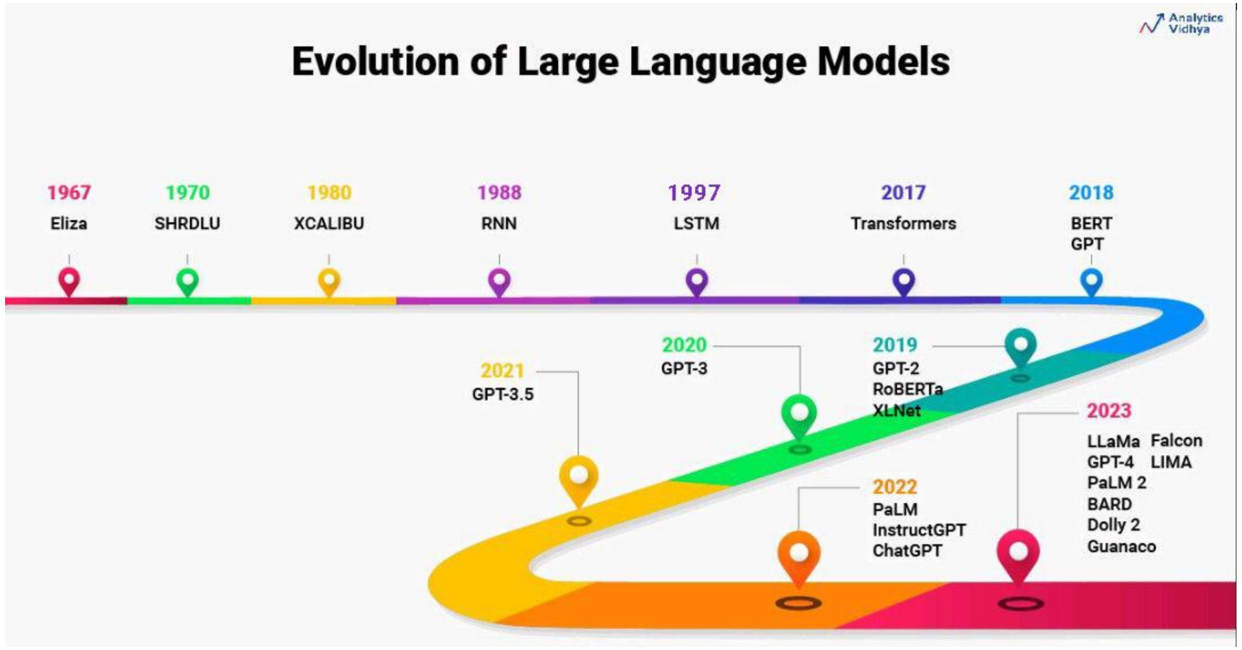
- Batched Gradient Descent
  - Batch Size = **N**
- Stochastic Gradient Descent
  - Batch Size = **1**
- Mini-batch Gradient Descent
  - Somewhere in the middle
  - Common:
    - **Batch Size** = 64, 128, 256, etc.
- Finding the optimal batch size will yield the fastest learning.

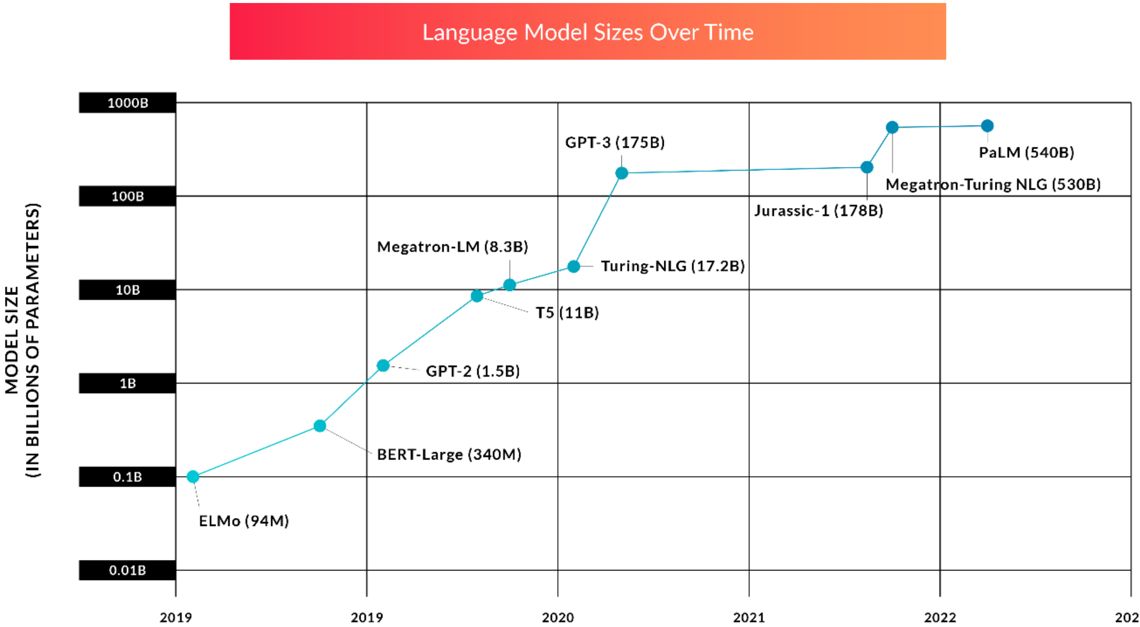


One full pass over ***N*** *is called an* ***epoch*** *of training*

Courtesy: https://www.jeremyjordan.me/gradient-descent/

Evolution of Language Models

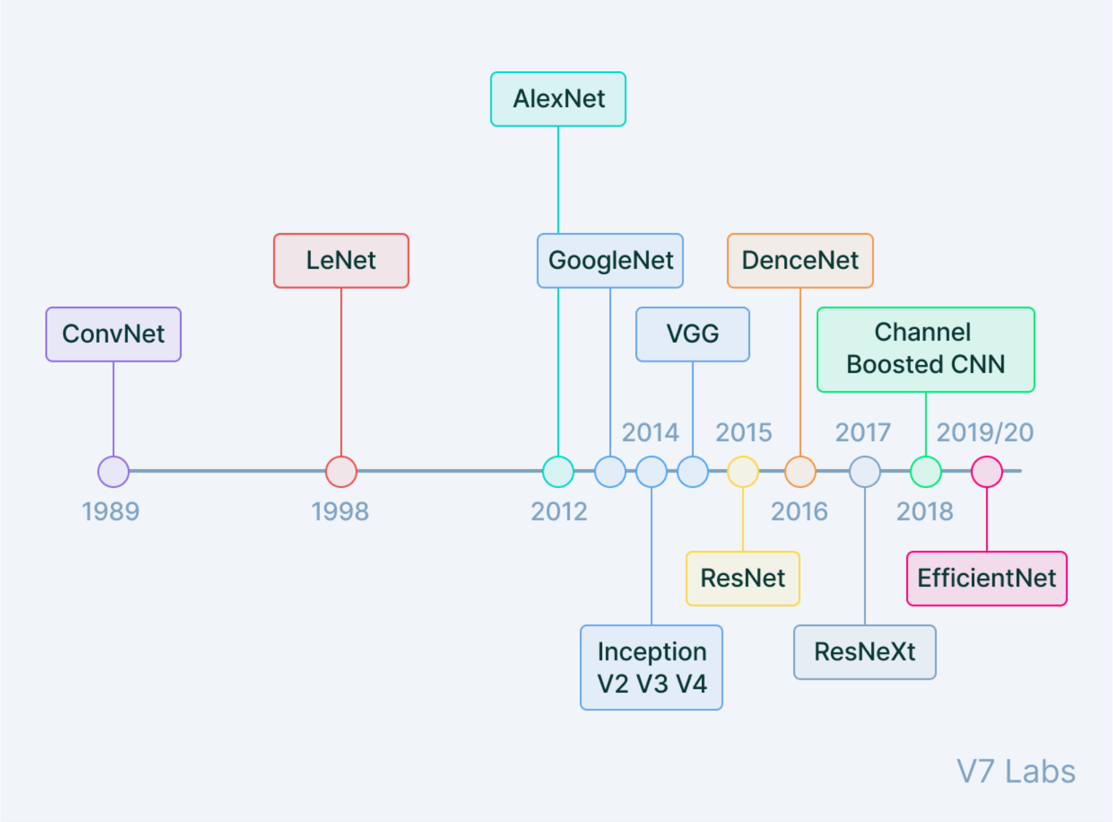




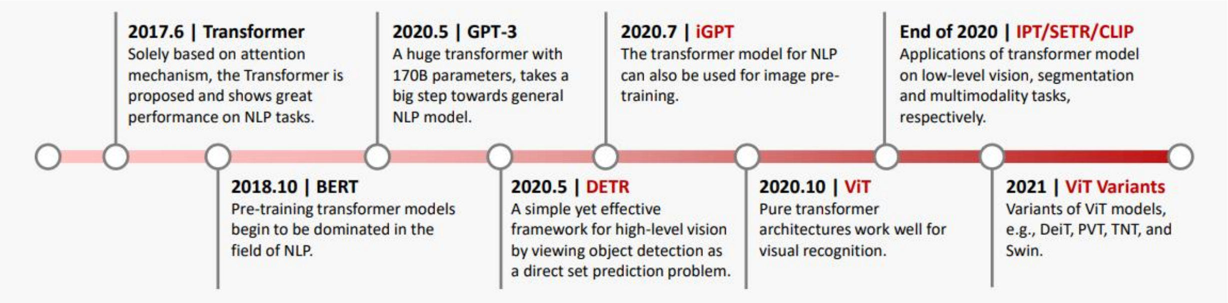
Courtesy: https://www.analyticsvidhya.com/blog/2023/07/build-your-own-large-language-models/
https://www.vinayiyengar.com/2022/08/04/the-promise-and-perils-of-large-language-models/

Evolution of Computer Vision Models

1. CNN Architectures



2. Vision Transformer Architectures



2017.6 | Transformer
Solely based on attention mechanism, the Transformer is proposed and shows great performance on NLP tasks.

2018.10 | BERT
Pre-training transformer models begin to be dominated in the field of NLP.

2020.5 | GPT-3
A huge transformer with 170B parameters, takes a big step towards general NLP model.

2020.5 | DETR
A simple yet effective framework for high-level vision by viewing object detection as a direct set prediction problem.

2020.7 | iGPT
The transformer model for NLP can also be used for image pre-training.

2020.10 | ViT
Pure transformer architectures work well for visual recognition.

End of 2020 | IPT/SETR/CLIP
Applications of transformer model on low-level vision, segmentation and multimodality tasks, respectively.

2021 | ViT Variants
Variants of ViT models, e.g., DeiT, PVT, TNT, and Swin.

Courtesy: https://www.v7labs.com/blog/convolutional-neural-networks-guide
A Survey on Vision Transformer (Kai Han et. Al 2022) https://arxiv.org/abs/2012.12556

Outline

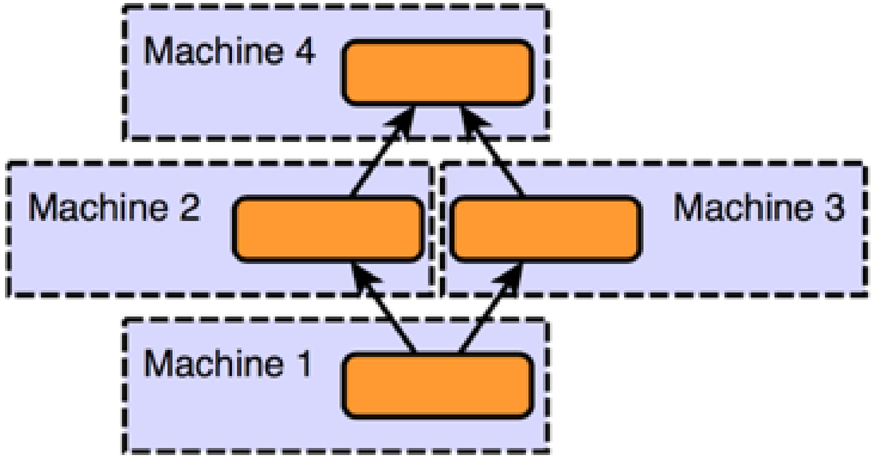
- Introduction
- **Training**
  - Deep Neural Network Training
  - **Distributed Deep Learning**
    - **Data Parallelism**
    - **Sharded Data Parallelism**
  - Training Solutions
- Inference
  - Tensor Parallelism
  - Expert Parallelism
- Conclusion

The Need for Parallel and Distributed Training

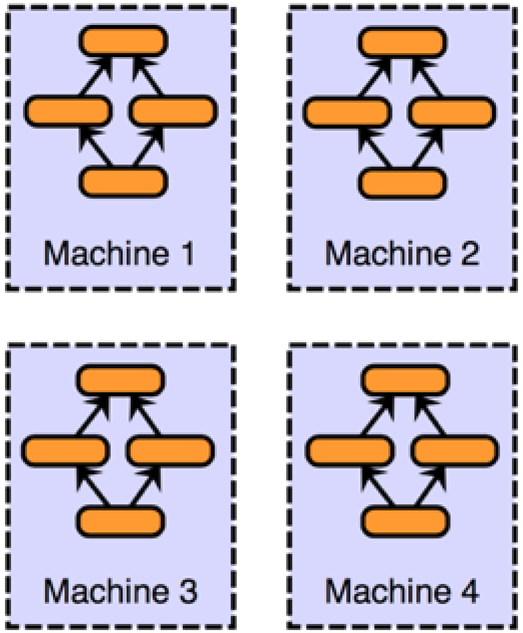
- Why do we need Parallel Training?
- Larger and Deeper models are being proposed
  - **Language Models: RNNs -> Transformers -> BERT – GPT – LLaMA**
  - **Vision Models: AlexNet** -> **ResNet** -> **NASNet – AmoebaNet → Vision Transformers**
  - DNNs require a lot of memory and a lot of computation
  - Larger models cannot fit a GPU's memory
- Single GPU training cannot keep up with ever-larger models
- Community has moved to multi-GPU training
- Multi-GPU in one node is good but there is a limit to Scale-up (8-16 GPUs)
- **Multi-node (Distributed or Parallel) Training is necessary!!**

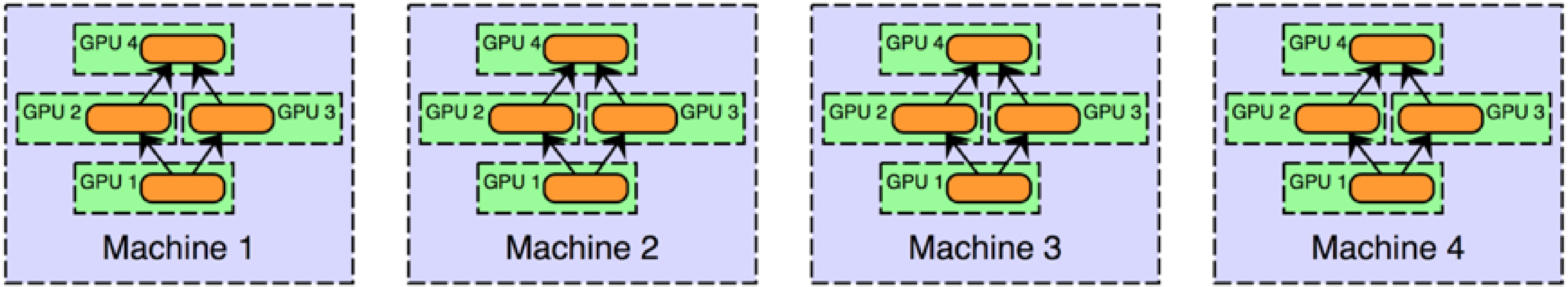
Parallelization Strategies

- Some parallelization strategies..
  - Data Parallelism or Model Parallelism
  - Hybrid Parallelism



Model Parallelism





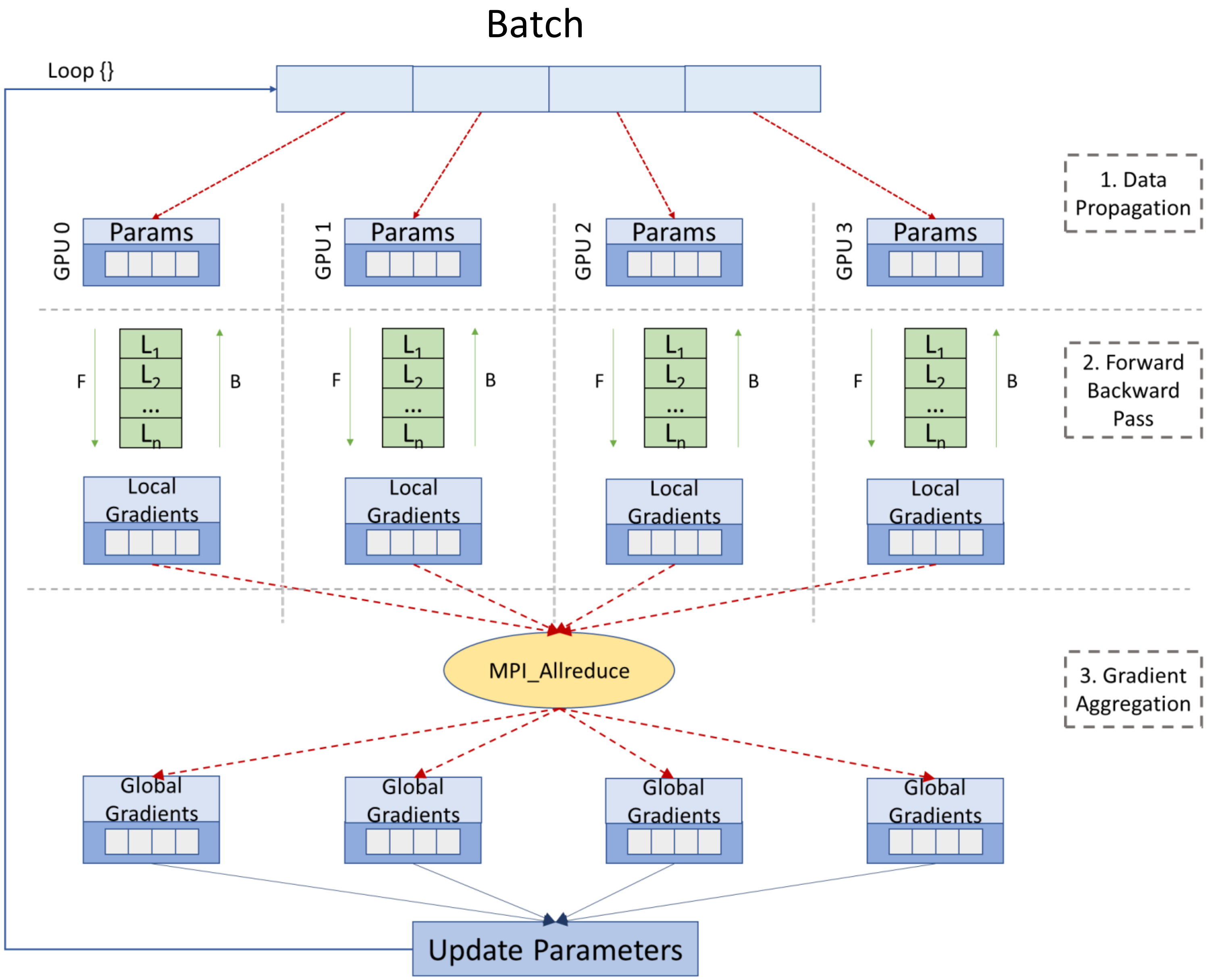
Hybrid (Model and Data) Parallelism

Data Parallelism

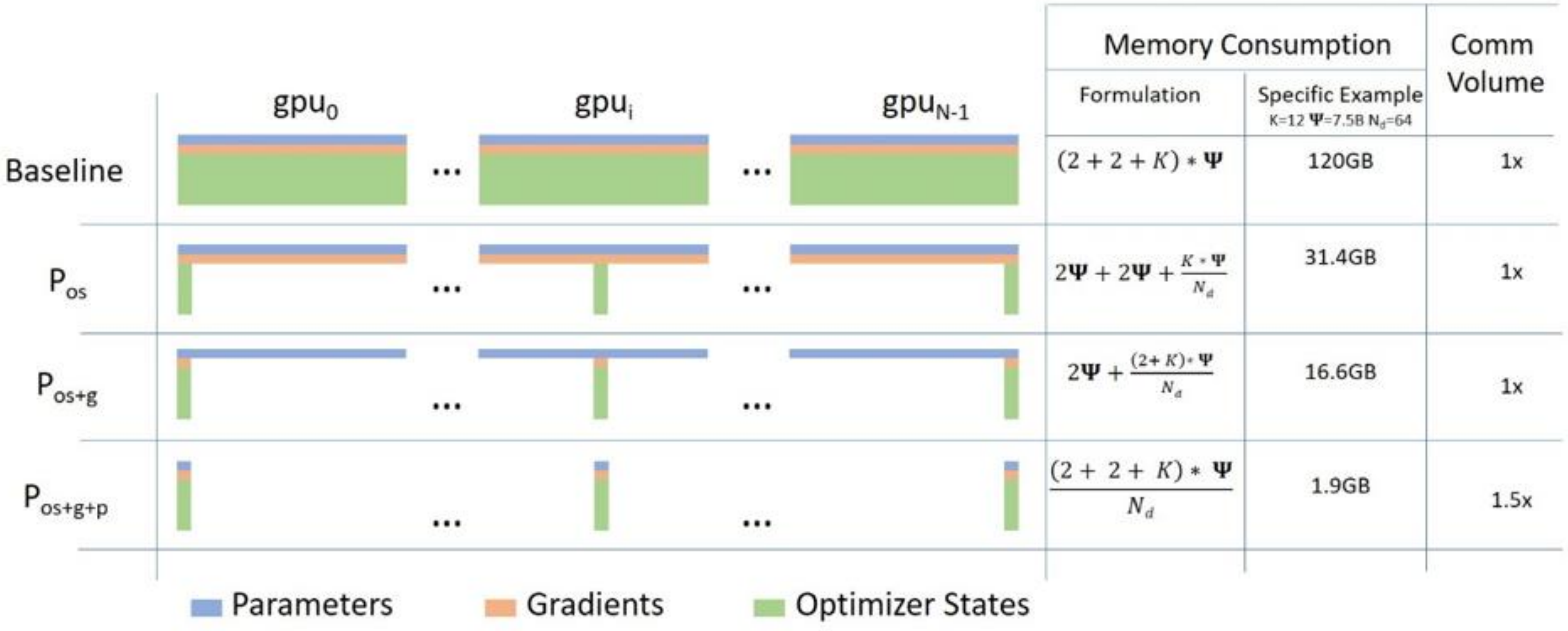
Courtesy: http://engineering.skymind.io/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks

Data Parallelism and MPI Collectives

- Step1: Data Propagation
  - Distribute the Data among GPUs
- Step2: Forward Backward Pass
  - Perform forward pass and calculate the prediction
  - Calculate Error by comparing prediction with actual output
  - Perform backward pass and calculate gradients
- Step3: Gradient Aggregation
  - Call MPI_Allreduce to reduce the local gradients
  - Update parameters locally using global gradients



Sharded Data Parallelism (DeepSpeed ZeRO)



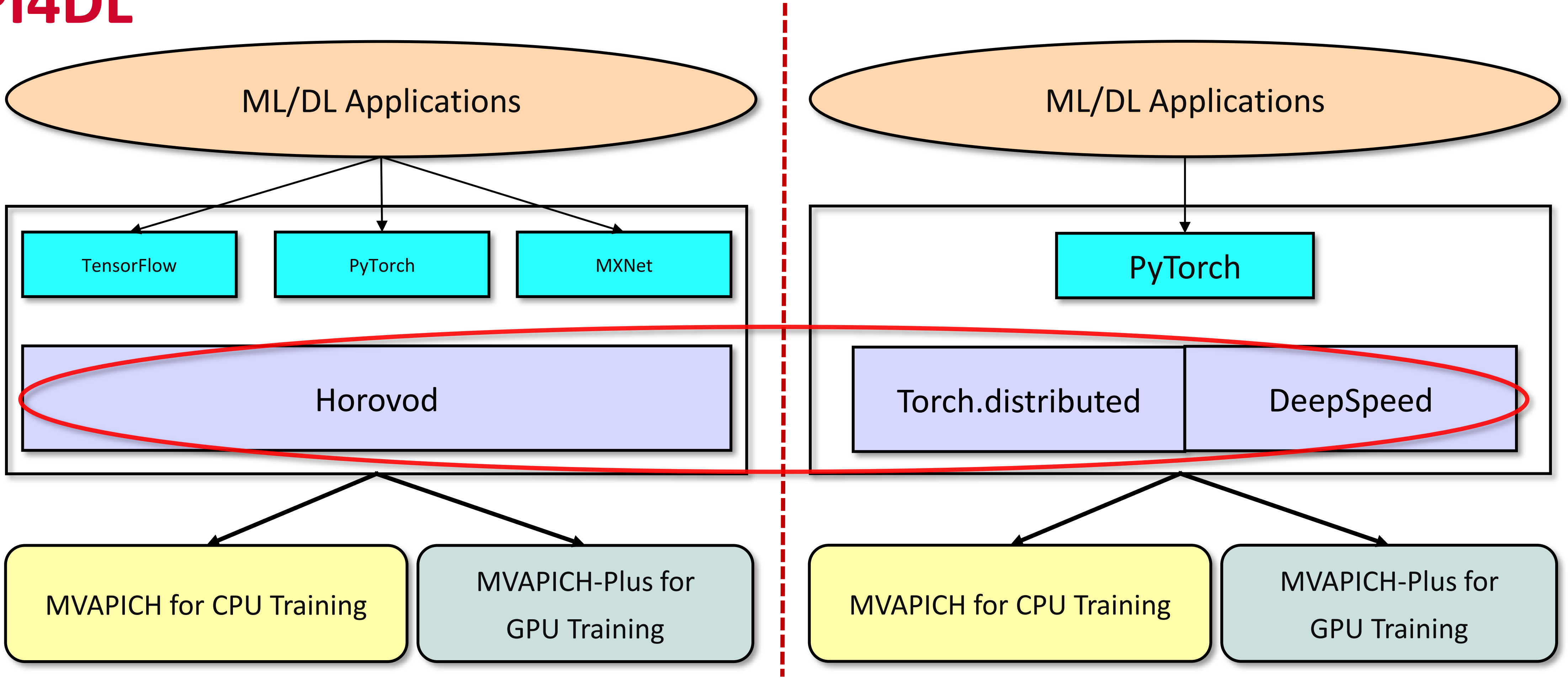
	Memory Consumption		Comm Volume
	Formulation	Specific Example K=12 Ψ=7.5B N_d=64	
Baseline	$(2+2+K)*\Psi$	120GB	1x
P_{os}	$2\Psi+2\Psi+\frac{K*\Psi}{N_d}$	31.4GB	1x
P_{os+g}	$2\Psi+\frac{(2+K)*\Psi}{N_d}$	16.6GB	1x
P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB	1.5x

- Instead of being limited by the **device** memory, we are now limited by the **aggregate** memory
- ZeRO-Infinity introduces offload to CPU memory or NVMe disk for the truly desperate
- Since ZeRO removes the DP memory limit, do we still need MP?
  - There are still models and data samples that don’t fit inside GPU memory ***even with ZeRO***
  - We can use pipeline + tensor parallelism along with ZeRO for these cases.

Outline

- Introduction
- **Training**
  - Deep Neural Network Training
  - Distributed Deep Learning
    - Data Parallelism
    - Sharded Data Parallelism
  - **Training Solutions**
- Inference
  - Tensor Parallelism
  - Expert Parallelism
- Conclusion

MVAPICH (MPI)-driven Infrastructure for ML/DL Training: MPI4DL



More details available from: https://github.com/OSU-Nowlab/pytorch/tree/hidl-2.0 and http://hidl.cse.ohio-state.edu

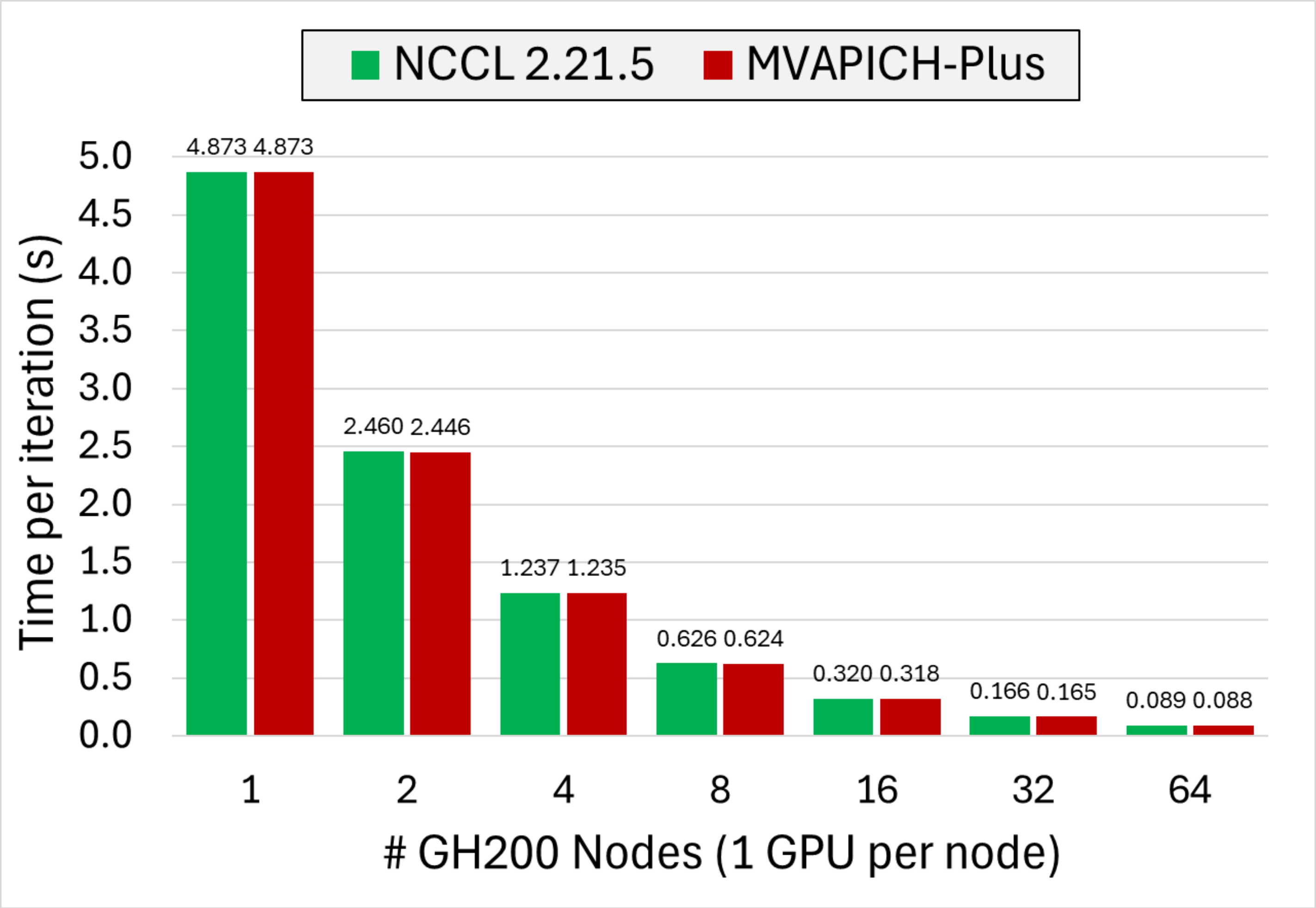
HiDL 2.0 Release

- Support for PyTorch 2.7.1 and later versions
- Full support for PyTorch Native DDP training
- Support for optimized MPI communication
  - Efficient large-message collectives (e.g., Allreduce) on various CPUs and GPUs
  - GPU-Direct Ring and Two-level multi-leader algorithms for Allreduce operations
  - Support for fork safety in distributed training environments
  - Exploits efficient large message collectives in MVAPICH-Plus 4.0 and later
- Open-source PyTorch version with advanced MPI backend support - Available in our PyTorch tag
- Vendor-neutral stack with competitive performance and throughput to GPU-based collective libraries
- Tested on modern HPC clusters (etc, OLCF Frontier, TACC Vista) with up-to-date accelerator generations (etc. AMD NVIDIA)
- Compatible with
  - InfiniBand Networks: Mellanox InfiniBand adapters (EDR, FDR, HDR, NDR)
  - Slingshot Networks: HPE Slingshot
  - GPU&CPU Support:
    - NVIDIA GPU A100, H100, GH200
    - AMD MI200 series GPUs
  - Software Stack:
    - CUDA [12.x] and Latest CuDNN
    - ROCm [6.x]
    - (NEW)PyTorch [2.7.1]
    - (NEW)Python [3.x]

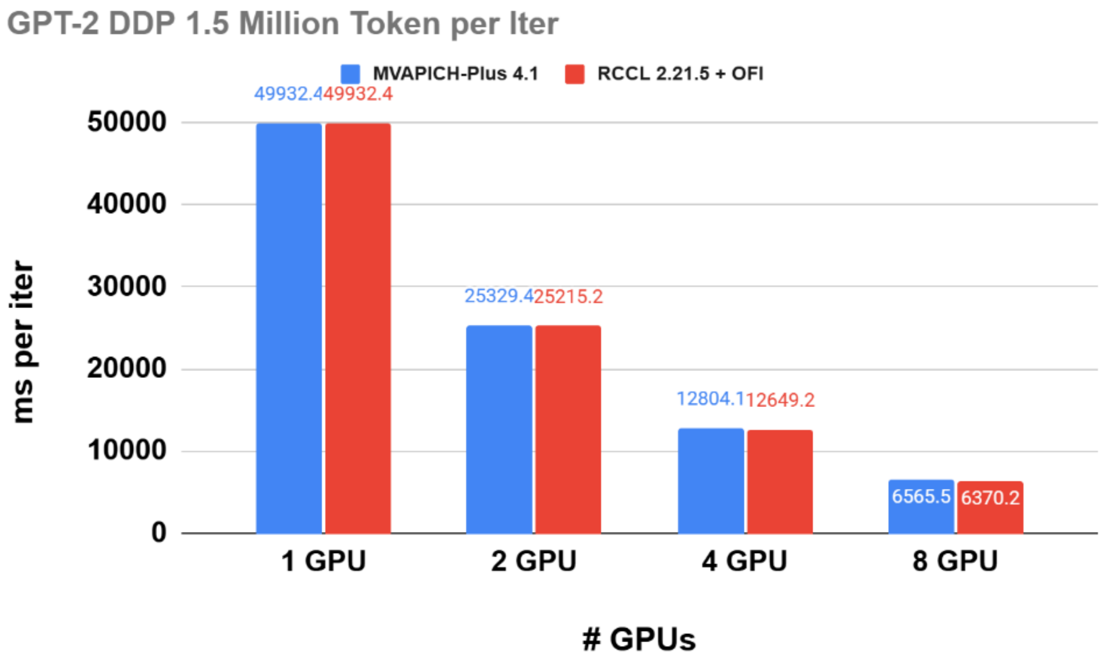
More details available from: https://github.com/OSU-Nowlab/pytorch/tree/hidl-2.0 and http://hidl.cse.ohio-state.edu

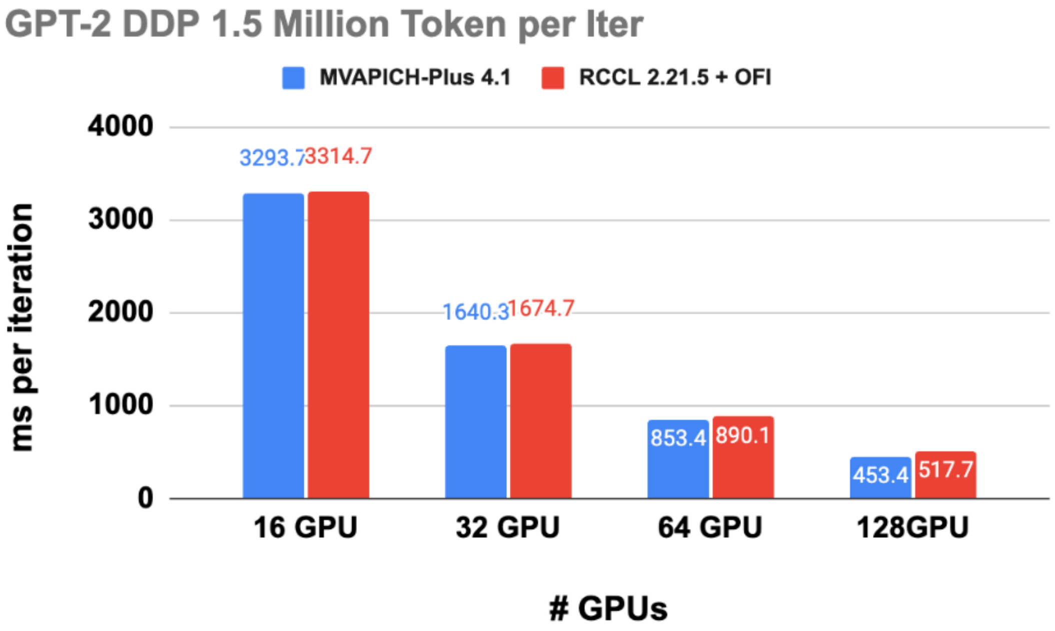
Distributed Data Parallel Training on GH200 (Vista)

- Torch Distributed
- Application: GPT-2 model training using nanoGPT.
- Hardware: Vista System @TACC
  - GH200 Superchips each with:
    - 72 ARM cores with 120 GB LPDDR.
    - H100 GPU with 96GB HBM3.
  - NVIDIA NDR InfiniBand (400Gb/s)
- Software:
  - PyTorch 2.6.0
  - NCCL 2.21.5
  - MVAPICH-Plus 4.1



Distributed Data Parallel Training (Frontier)

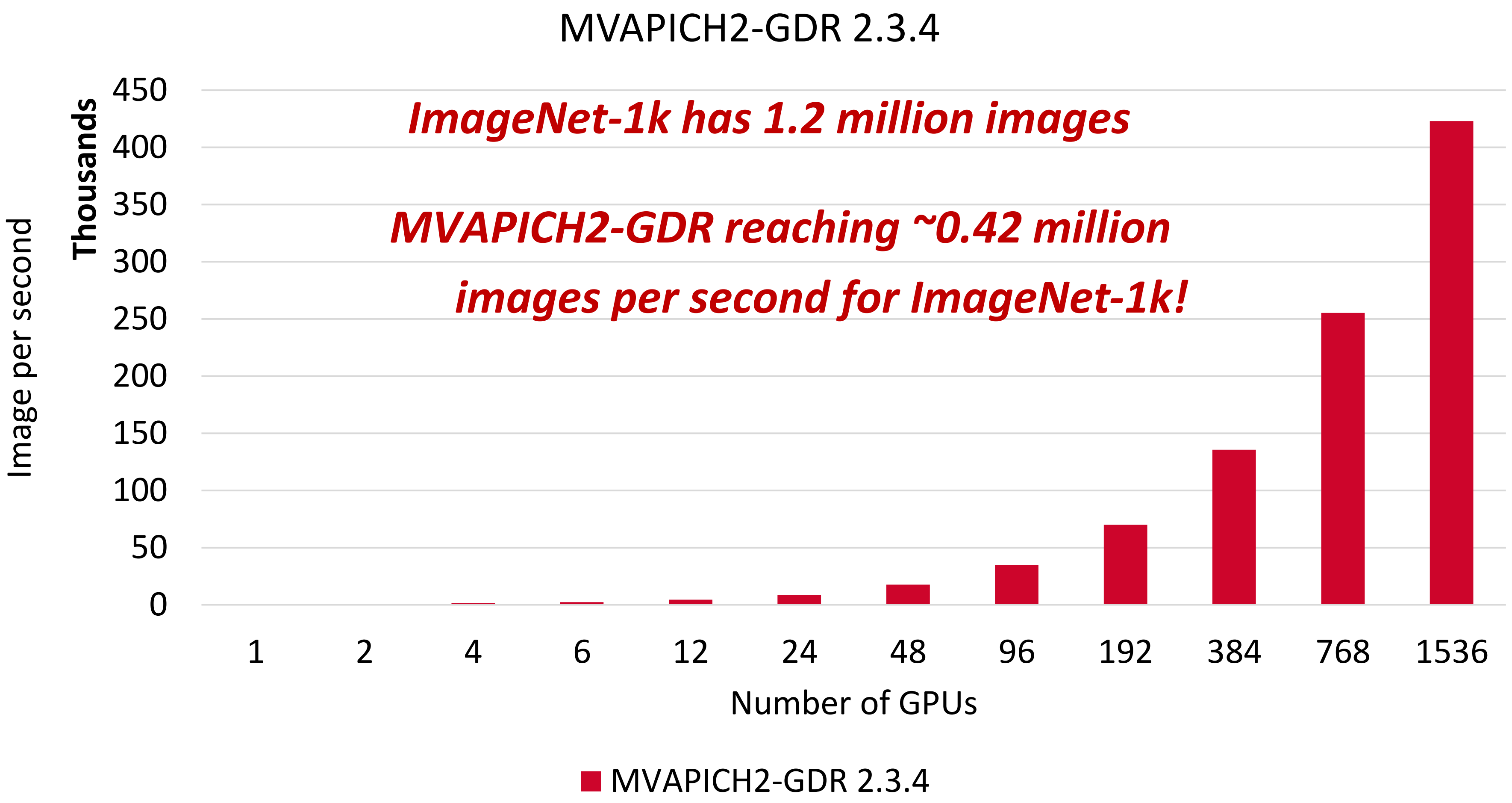




- End-to-end GPT-2 Training with Openwebtext using Distributed Data Parallel
- **12.4%** less ms per iteration (compared to RCCL 2.21.5 + OFI) for 128 GPUs

Distributed TensorFlow on ORNL Summit (1,536 GPUs)

- ResNet-50 Training using TensorFlow benchmark on SUMMIT -- 1536 Volta GPUs!
- 1,281,167 (1.2 mil.) images
- Time/epoch = 3 seconds
- Total Time (90 epochs) = 3 x 90 = 270 seconds = **4.5 minutes!**



ImageNet-1k has 1.2 million images

MVAPICH2-GDR reaching ~0.42 million images per second for ImageNet-1k!

Platform: The Summit Supercomputer (#2 on Top500.org) – 6 NVIDIA Volta GPUs per node connected with NVLink, CUDA 10.1

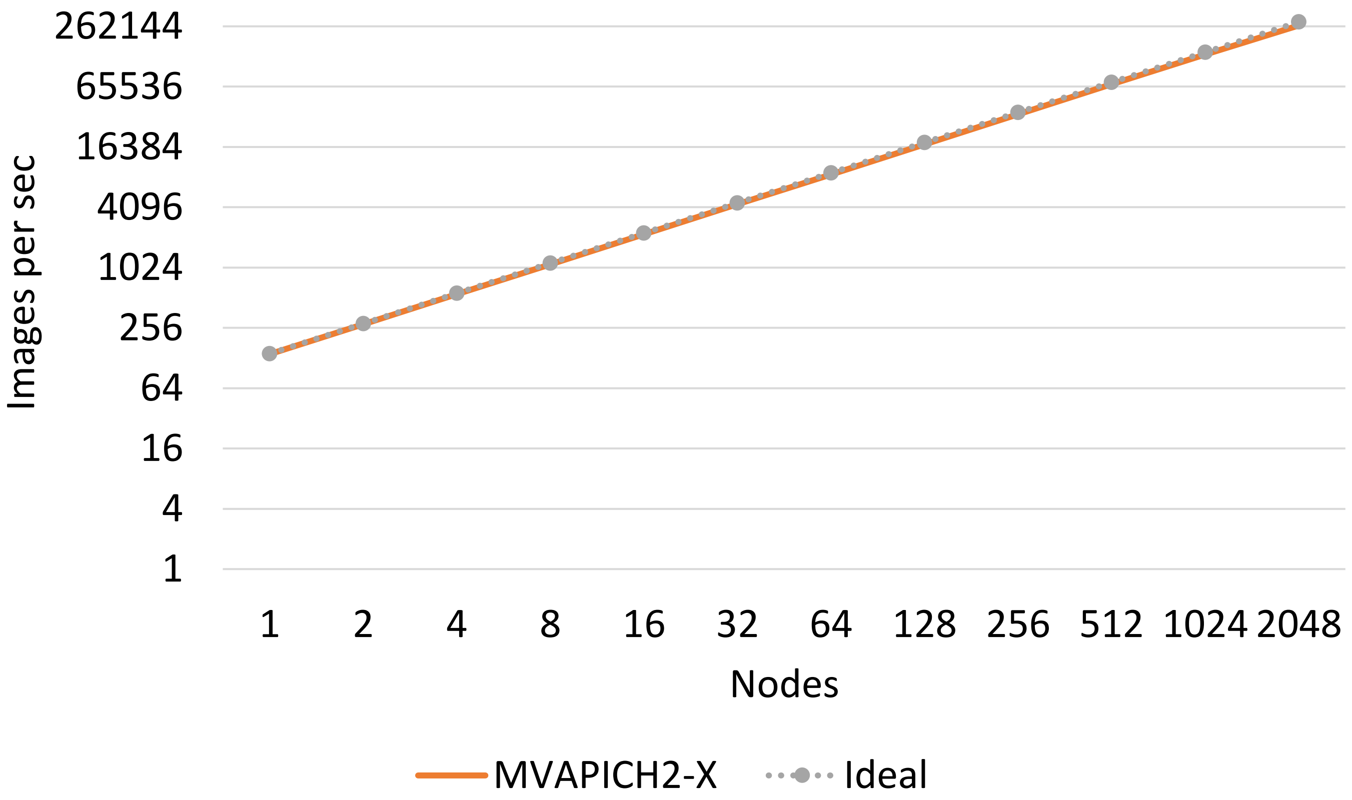
Distributed TensorFlow on TACC Frontera (2048 CPU nodes)

- Scaled TensorFlow to 2048 nodes on Frontera using MVAPICH2 and IntelMPI

- MVAPICH2 delivers close to the ideal performance for DNN training

- Report a peak of 260,000 images/sec on 2048 nodes

- On 2048 nodes, ResNet-50 can be trained in 7 minutes!

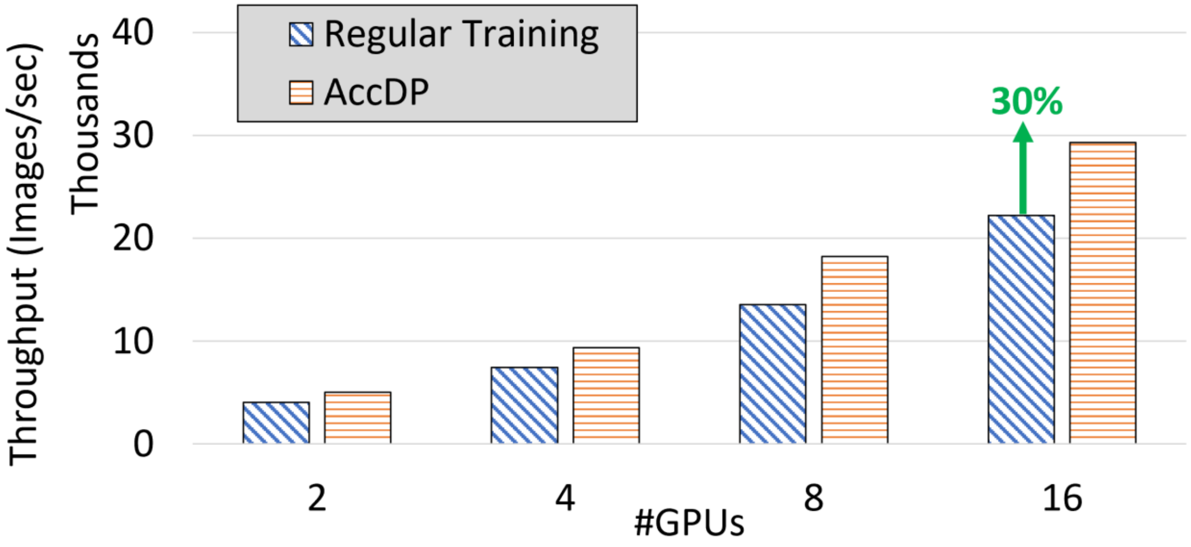


A. Jain, A. A. Awan, H. Subramoni, DK Panda, "Scaling TensorFlow, PyTorch, and MXNet using MVAPICH2 for High-Performance Deep Learning on Frontera", DLS '19 (SC '19 Workshop).

AccDP: Exploiting Data Parallelism

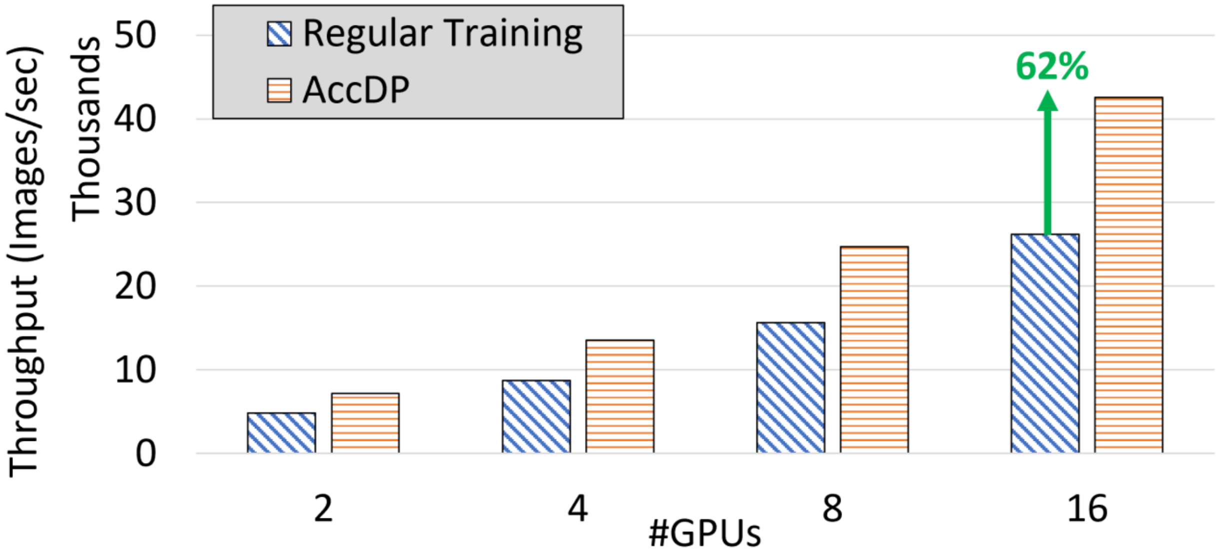
Multi node with ResNet18

- ResNet18 training throughput comparison between regular training and AccDP (proposed design) for different DNN models on up to 8 nodes 2 GPUs per node (16 GPUs) with 4 MPS clients per GPU



Multi node with ShuffleNet

- ShuffleNet training throughput comparison between regular training and AccDP (proposed design) for different DNN models on up to 8 nodes 2 GPUs per node (16 GPUs) with 4 MPS clients per GPU.



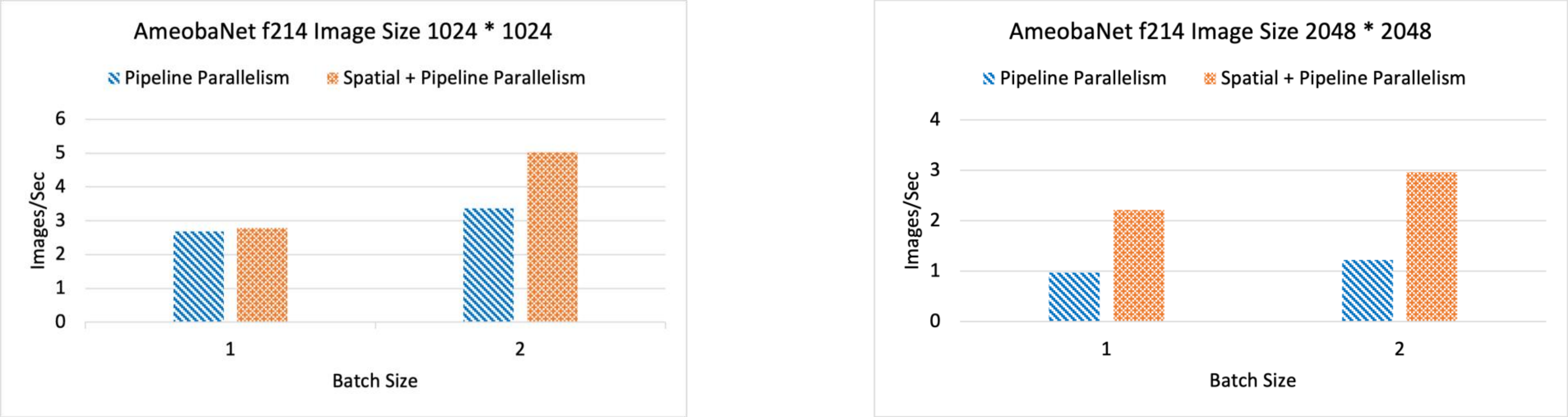
N. Alnaasan, A. Jain, A. Shafi, H. Subramoni, and DK Panda, “AccDP: Accelerated Data-Parallel Distributed DNN Training for Modern GPU-Based HPC Clusters”, HiPC’22.

MPI4DL v0.6

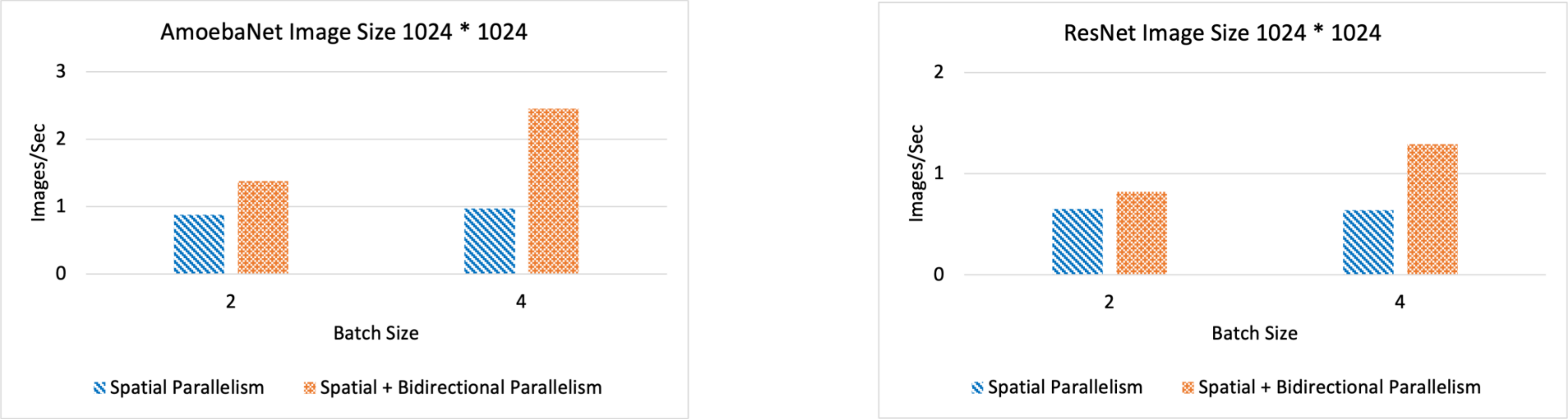
MPI4DL v0.6 is a distributed, accelerated and memory efficient training framework for very high-resolution images that integrates Spatial Parallelism, Bidirectional Parallelism, Layer Parallelism, and Pipeline Parallelism.

Features:

- Based on PyTorch
- Support for training very high-resolution images
- Distributed training support for:
  - Model Parallelism
    - Layer Parallelism (LP)
    - Pipeline Parallelism (PP)
  - Spatial Parallelism for High Resolution Images
    - Spatial and Layer Parallelism (SP+LP)
    - Spatial and Pipeline Parallelism (SP+PP)
  - Memory Efficient Bidirectional Parallelism (GEMS)
    - Bidirectional and Layer Parallelism (GEMS+LP)
    - Bidirectional and Pipeline Parallelism (GEMS+PP)
    - Spatial, Bidirectional and Layer Parallelism (SP+GEMS+LP)
    - Spatial, Bidirectional and Pipeline Parallelism (SP+GEMS+PP)
- Support for different image sizes and custom datasets.
- Exploits collective features of MVAPICH2-GDR



Throughput comparison of Pipeline Parallelism and Pipeline + Spatial Parallelism techniques for AmoebaNet on 1024 * 1024 and 2048 * 2048 image sizes.



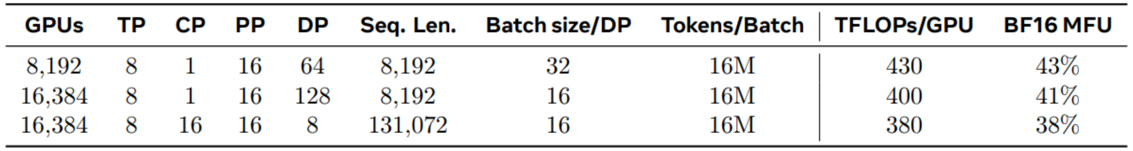
Throughput comparison of Spatial Parallelism and Spatial + Bidirectional Parallelism for AmoebaNet and ResNet with the following configurations: 5 model splits,4 spatial parts, and 2 model replicas for Bidirectional Parallelism.

Outline

- Introduction
- Training
  - Deep Neural Network Training
  - Distributed Deep Learning
    - Data Parallelism
    - Sharded Data Parallelism
  - Training Solutions
- Inference
  - Tensor Parallelism
  - Expert Parallelism
- Conclusion

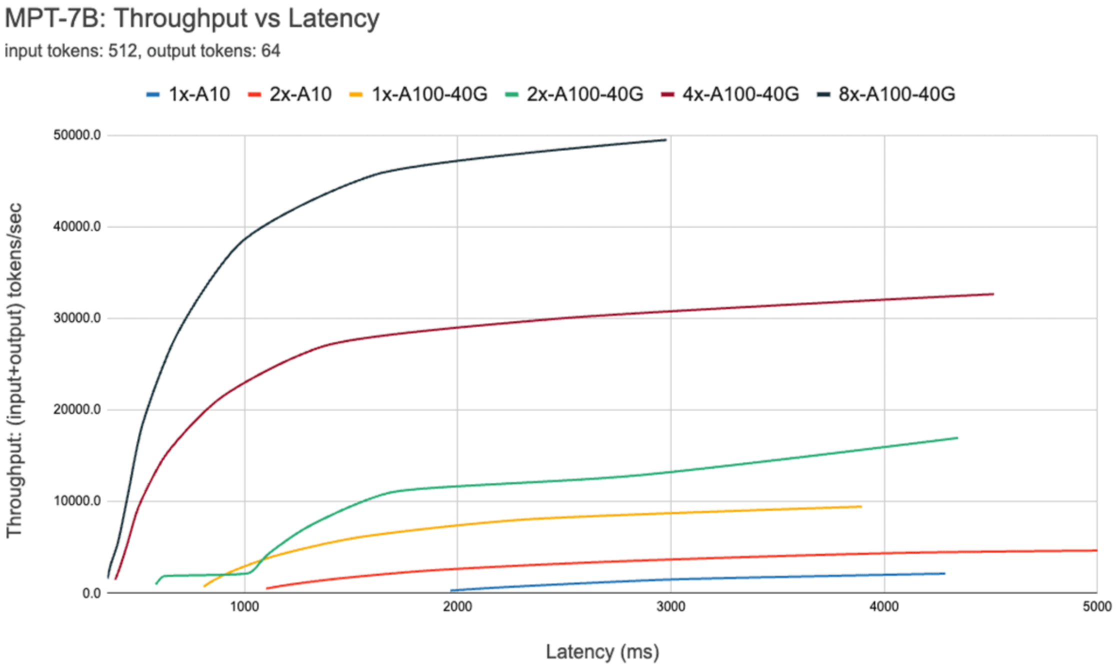
Introduction to Inference

- Training:
  - Key metrics: Throughput, batch size, MFU (Model FLOPs Utilization, i.e., hardware utilization) …
  - Example (LLaMA 3 405B pre-training) ->



GPUs	TP	CP	PP	DP	Seq. Len.	Batch size/DP	Tokens/Batch	TFLOPs/GPU	BF16 MFU
8,192	8	1	16	64	8,192	32	16M	430	43%
16,384	8	1	16	128	8,192	16	16M	400	41%
16,384	8	16	16	8	131,072	16	16M	380	38%

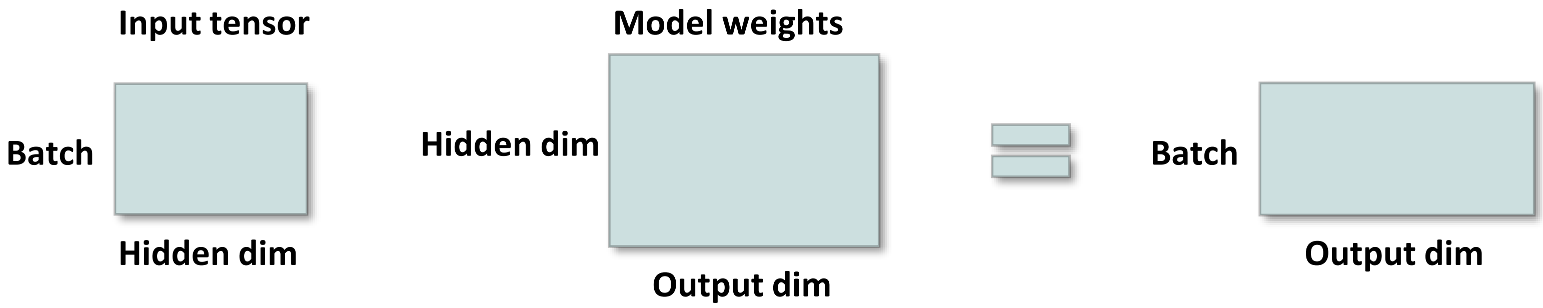
- Inference:
  - Optimization order: First minimize Latency, then maximize throughput.
  - LLM serving example --- Service Level Objectives (SLOs):
    - Time to First Token (TTFT): < 5 seconds
    - Time between Tokens (TBT): < 25 milliseconds
  - Goal: Without exceeding SLO thresholds, increase throughput as much as possible.



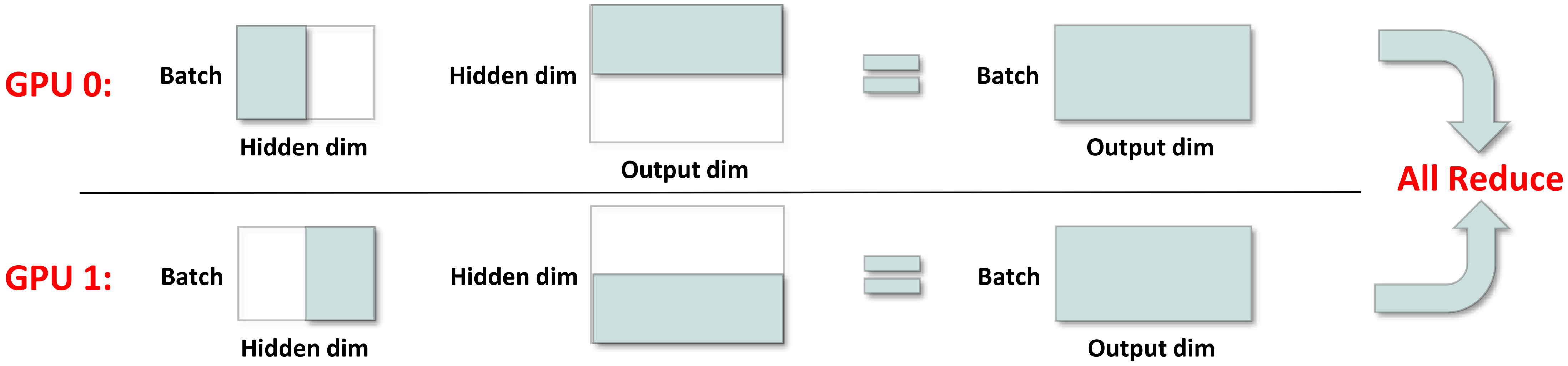
Let's see how to optimize latency!

Tensor Parallelism

- LLM models consist of matrix multiplications.

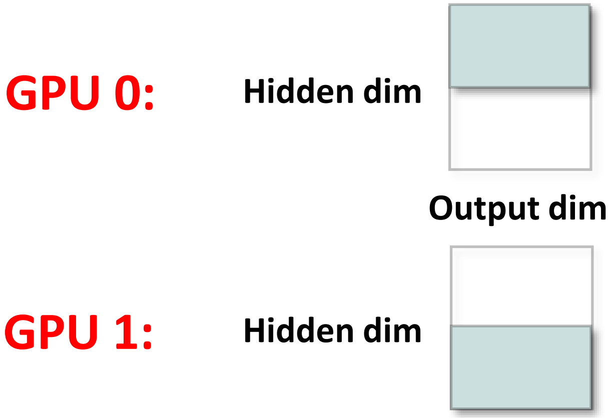


- Tensor Parallelism splits along hidden dim, and distributes the computation to multiple GPUs.



Tensor Parallelism

- Tensor Parallelism (TP) is widely used in both training and inference.
- For training, TP is used to distribute model weights and hence increase the batch size.
- For inference, TP is used to reduce the latency of loading model weights.

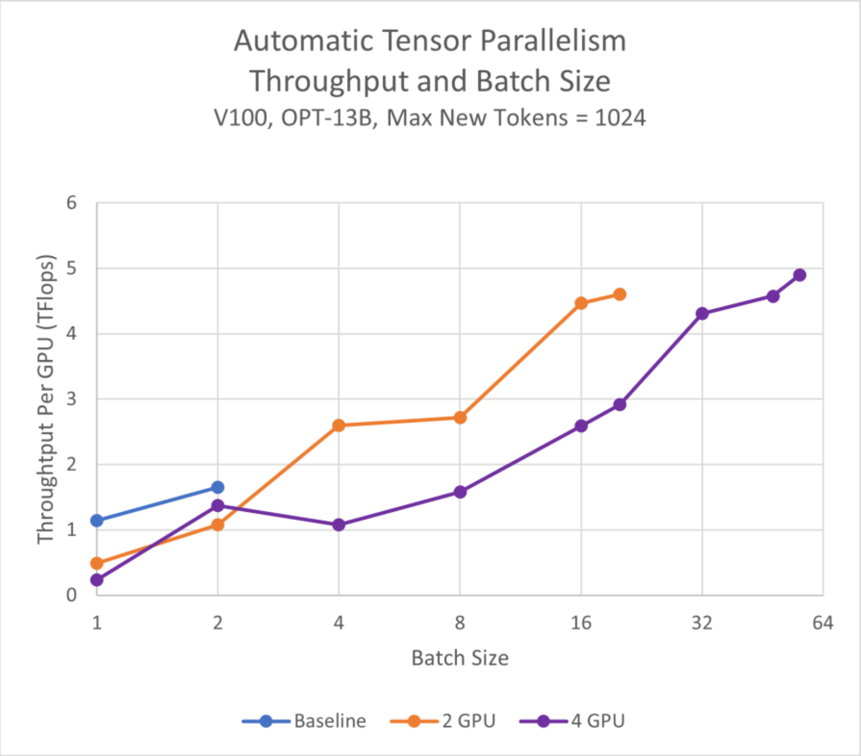

GPU 0:

Hidden dim

Output dim

GPU 1:

Hidden dim

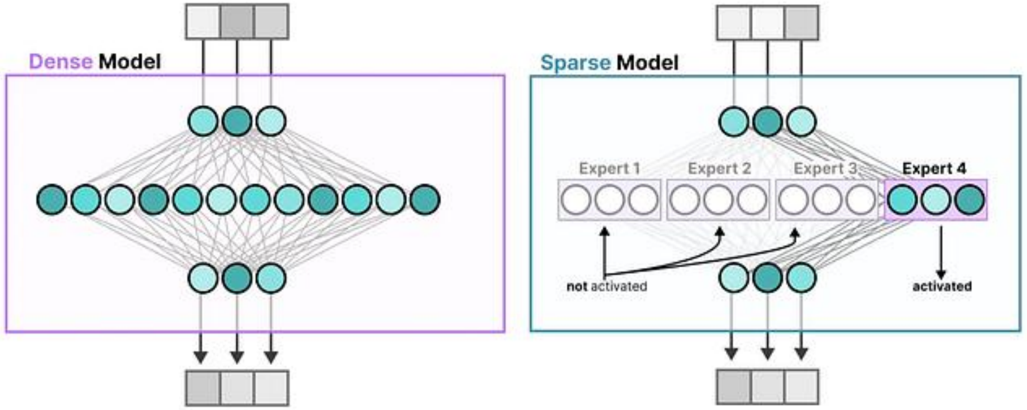

Automatic Tensor Parallelism
Throughput and Batch Size
V100, OPT-13B, Max New Tokens = 1024

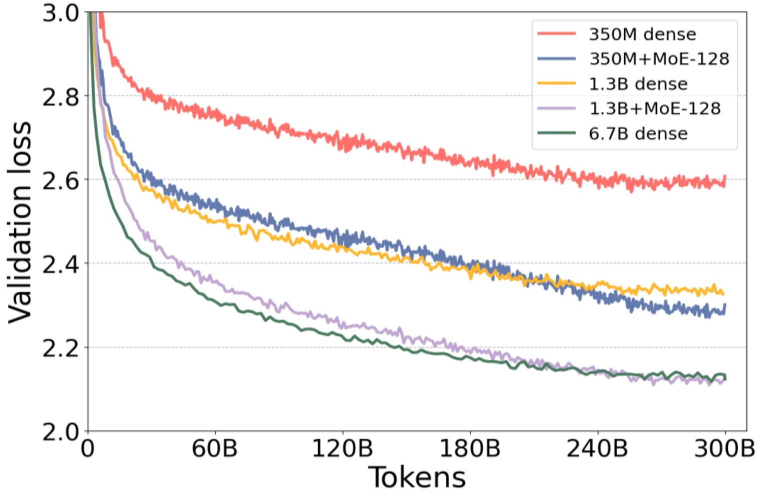
Outline

- Introduction
- Training
  - Deep Neural Network Training
  - Distributed Deep Learning
    - Data Parallelism
    - Sharded Data Parallelism
  - Training Solutions
- Inference
  - Tensor Parallelism
  - Expert Parallelism
- Conclusion

Mixture of Experts

- Mixture of Expert (MoE)
  - MoE in LLM was first proposed by Google in 2021.
  - In 2025, MoE is widely used in almost ALL leading LLMs
    - GPT-4/5, DeepSeek V3/R1, Qwen 3, LLaMA 4, …
- Sparsity in MoE:
  - Compared to dense models, MoE is much more sparse
  - Experts can diverse to different tasks, improving the performance.

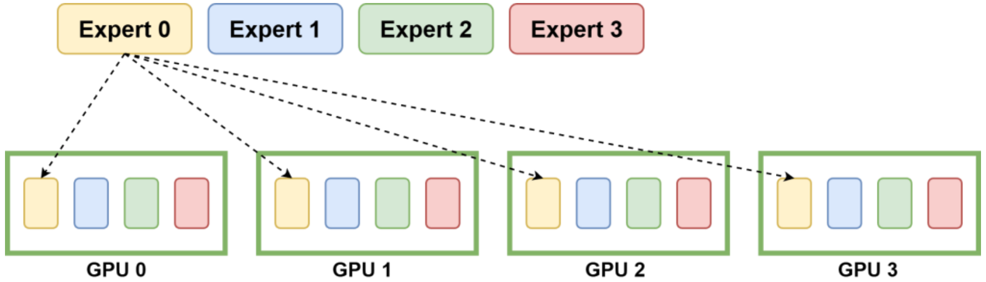




Tensor Parallel in MoE

- There are two ways to distribute MoE models.

- Tensor Parallel

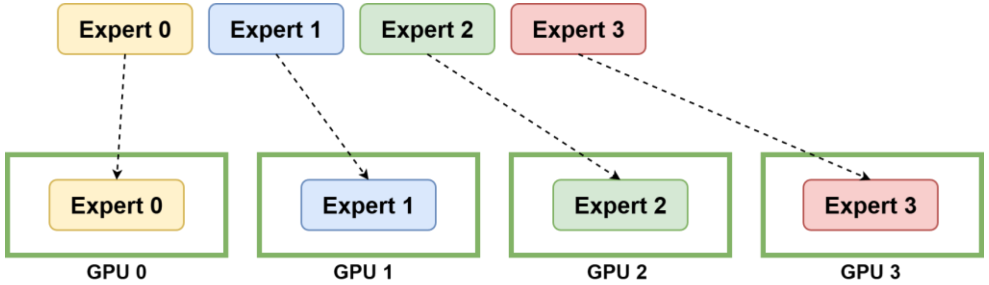


- Each GPU will hold a shard of each expert.
  - Given an input token of size [1, d], and each expert is of size [d, d]
  - As we use AllReduce in TP, the overall communication volume is

$$4x[1, d]$$

Expert Parallel in MoE

- Expert Parallel



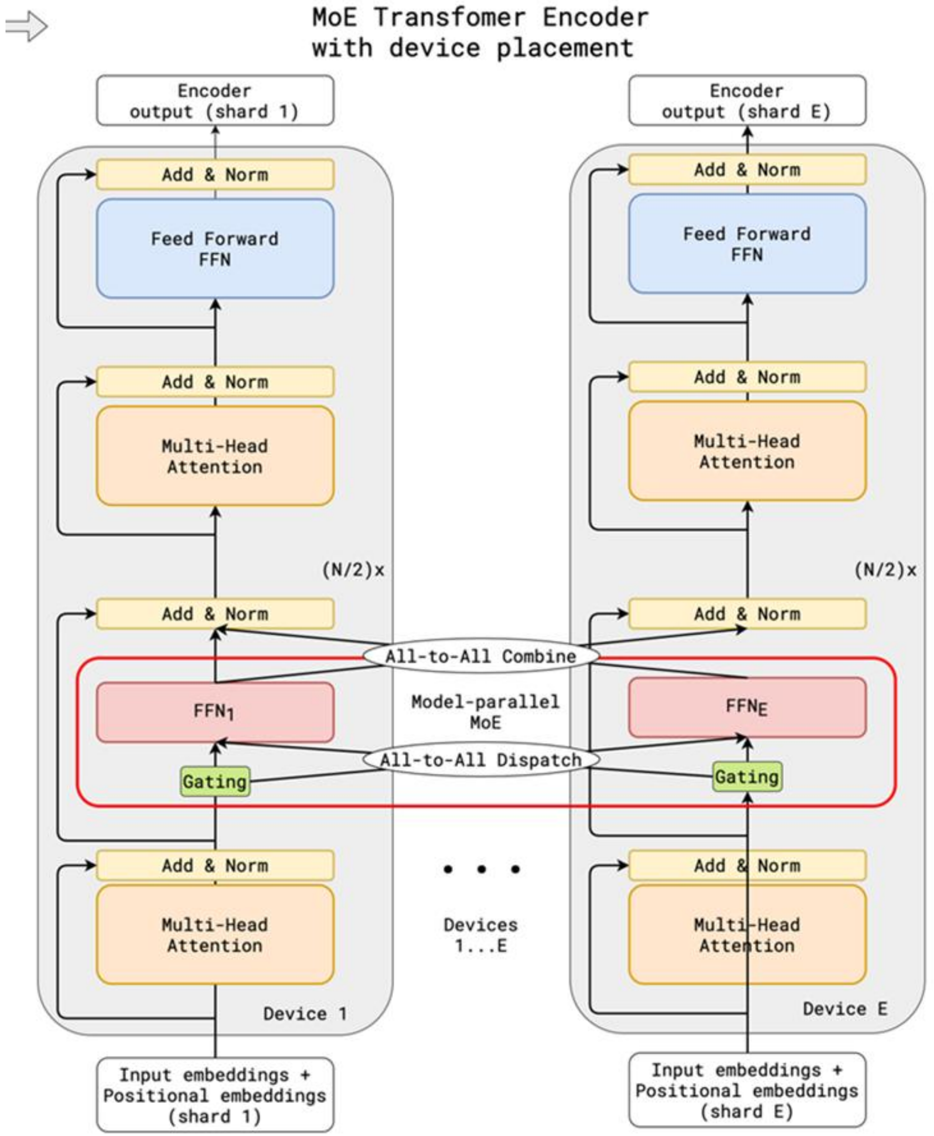
- Each GPU will hold one expert completely.
  - Given an input token of size [1, d], and each expert is of size [d, d]
  - In EP, the overall communication volume is

2x[1, d]

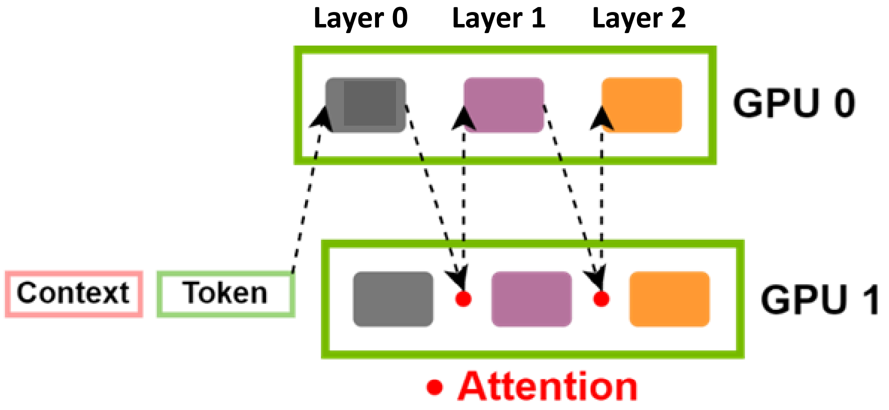
Compared to TP: 4x[1, d]

Num. of GPUs	TP	EP
4	4x[1,d]	2x[1,d]
8	8x[1,d]	2x[1,d]
16	16x[1,d]	2x[1,d]

Alltoall in Expert Parallel



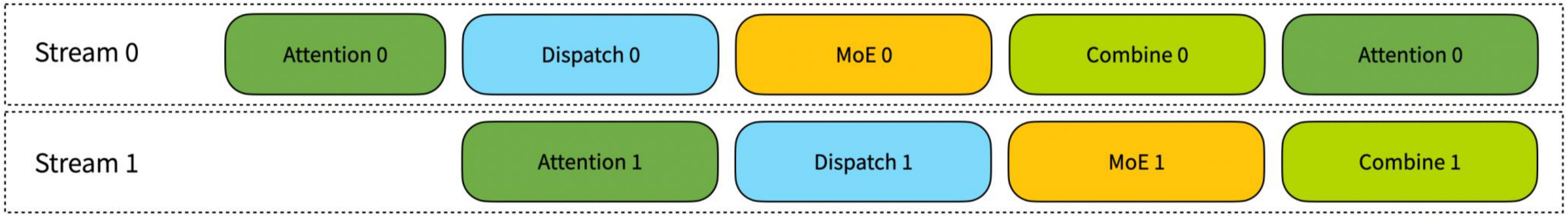
- In MoE, we need two Alltoall.
  - All-to-All Dispatch: tokens go to experts
  - All-to-All Combine: tokens go back to the original GPU.



Overlapping communication with computation

- During the two All-to-all communication, other operations will be blocked.
- A simple solution is pipelining.
- Assume we have 2 batches,
  - Perform operations per batch, and overlap different stages with the other batch.

Traditional overlapping with communication SMs

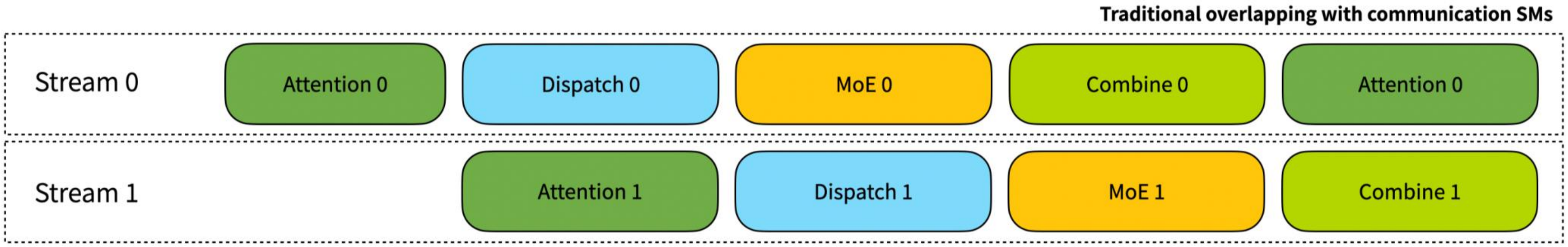


Stream 0: Attention 0, Dispatch 0, MoE 0, Combine 0, Attention 0

Stream 1: Attention 1, Dispatch 1, MoE 1, Combine 1

Issue: GPU communication libraries use SM to perform the RDMA read/write.

Overlapping communication with computation

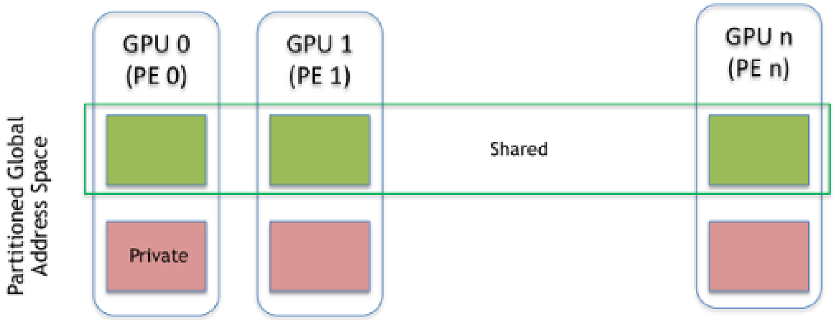


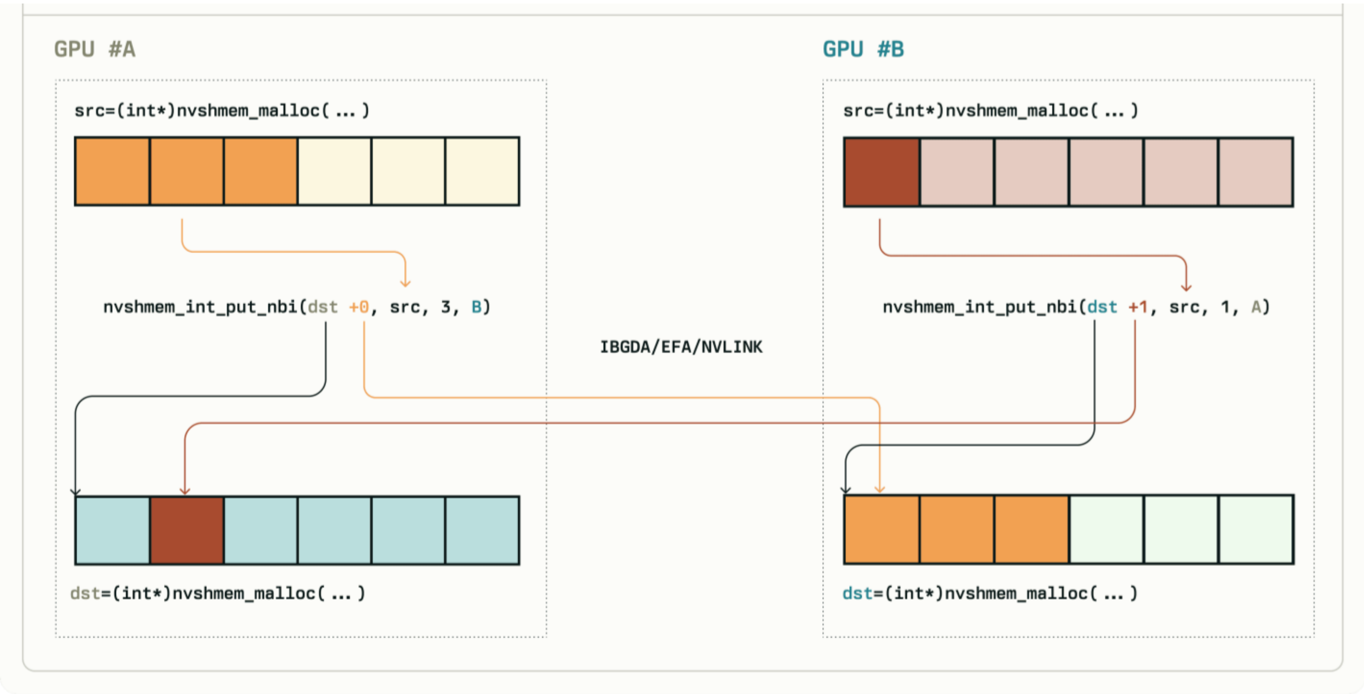
What's more?

1. For example, NCCL defaultly reserves up to 32/64 thread blocks for communication.
2. We have less SM for computation.
3. On H100, each SM has CUDA cores and Tensor cores,
    1. Computation uses both CUDA cores and (mostly)Tensor cores.
    2. Communication only uses CUDA cores.

All-to-all with InfiniBand GPUDirect Async (IBGDA)

- First introduced in DeepSeek R1 (March 2025)
- Leverage NVSHMEM
  - Requires symmetric memory allocation
  - Allows RDMA get/put inside a GPU kernel function



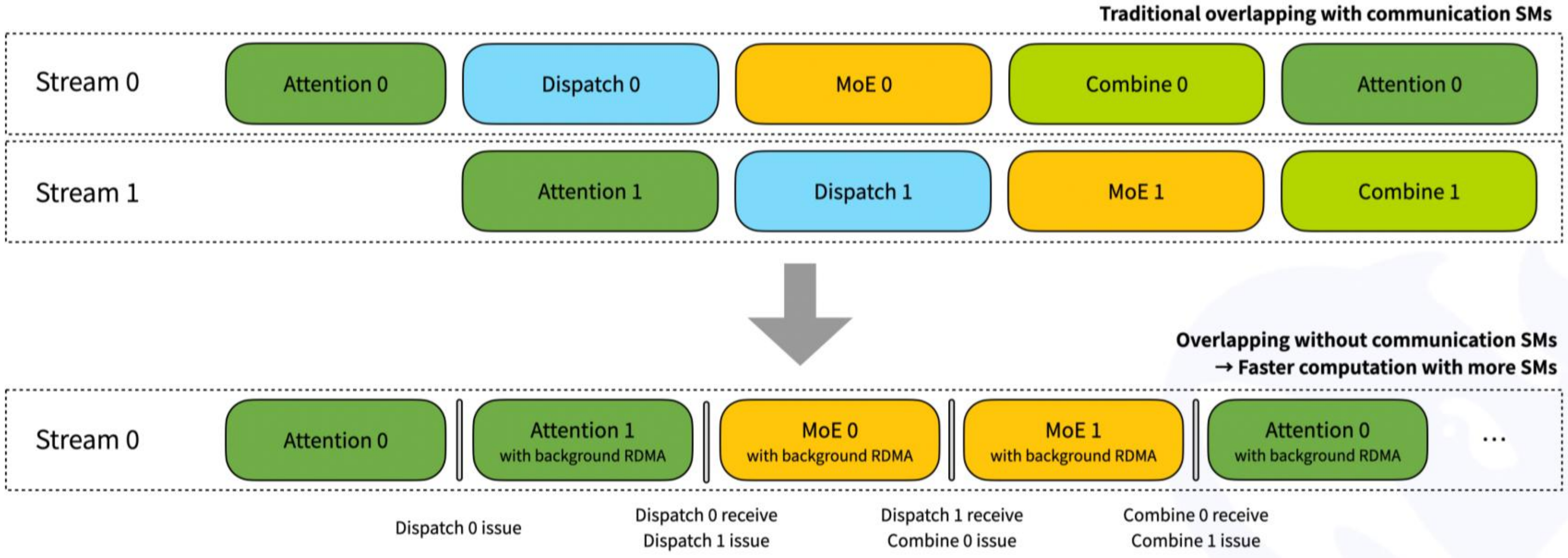


```
src=(int*)nvshmem_malloc( ... )
nvshmem_int_put_nbi(dst +0, src, 3, B)
dst=(int*)nvshmem_malloc( ... )
```

```
src=(int*)nvshmem_malloc( ... )
nvshmem_int_put_nbi(dst +1, src, 1, A)
dst=(int*)nvshmem_malloc( ... )
```

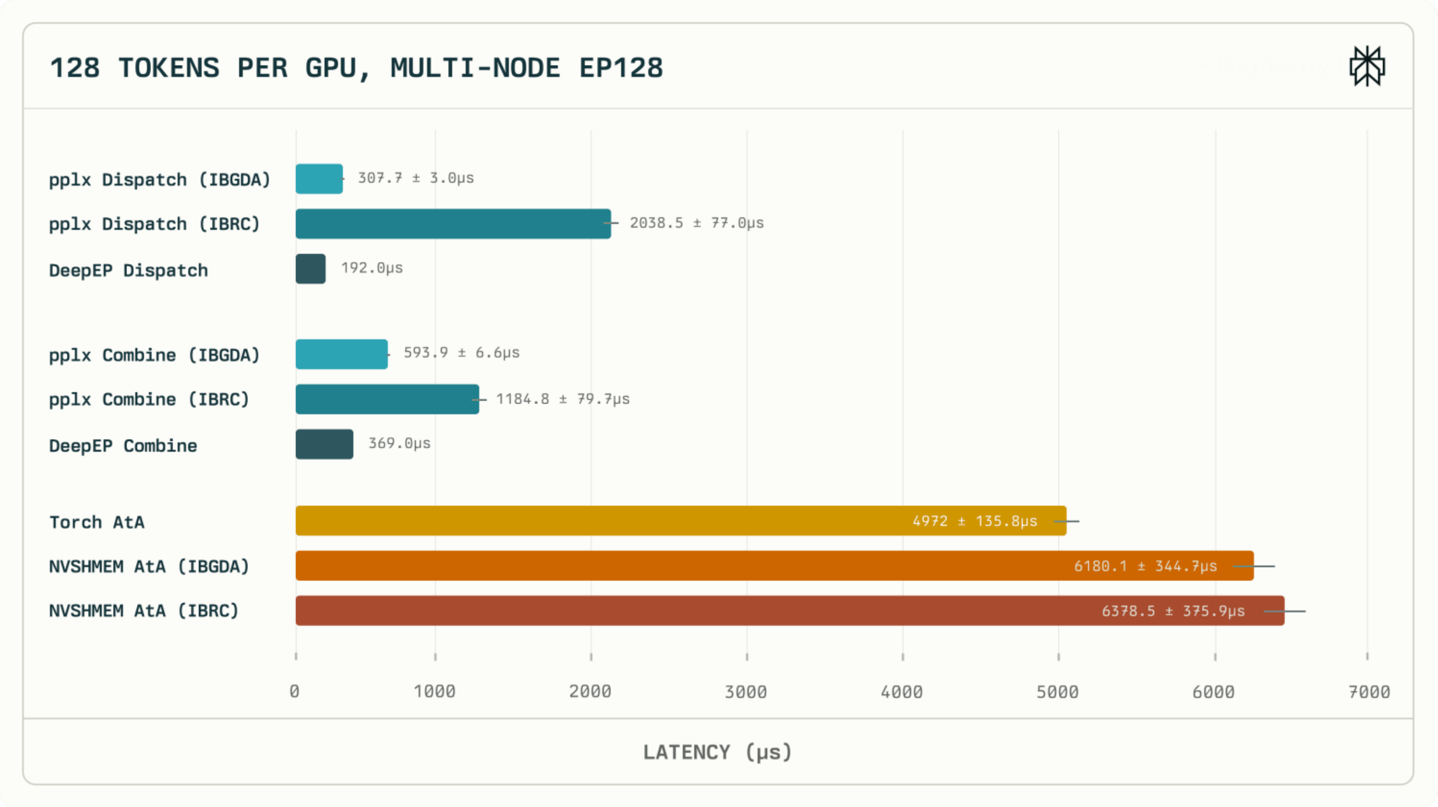
All-to-all InfiniBand GPUDirect Async (IBGDA)

- Fine-grained overlapping
  - All SMs are used for computation.
  - Within each thread block, threads are splitted to perform computation-RDMA overlapping.



More on IBGDA in MoE

- 128 GPUs, 16 nodes
- InfiniBand with CX-7 NIC



https://www.perplexity.ai/hub/blog/efficient-and-portable-mixture-of-experts-communication

Outline

- Introduction
- Training
  - Deep Neural Network Training
  - Distributed Deep Learning
    - Data Parallelism
    - Sharded Data Parallelism
  - Training Solutions
- Inference
  - Tensor Parallelism
  - Expert Parallelism
- Conclusion

Conclusion

- Exponential growth in in Deep Learning frameworks and workloads.
- Provided an overview of issues, challenges, and opportunities for designing efficient communication runtimes for training and inference
  - Efficient, scalable, and hierarchical designs are crucial for DL frameworks
  - Co-design of communication runtimes and DL frameworks will be essential
- Presented a set of state-of-the-art solutions to demonstrate the complex interaction between DL middleware with the underling HPC technologies and middleware
- Demonstrated the scalability of efficient solutions on a diverse set of DL workloads.

Acknowledgments to all the Heroes (Past/Current Students and Staffs)

Current Students (Under/Graduate)

- N. Alnaasan (Ph.D.)
- Q. Anthony (Ph.D.)
- C.-C. Chen (Ph.D.)
- T. Chen (Ph.D.)
- N. Contini (Ph.D.)
- S. Gumaste (Ph.D.)
- J. Hatef (Ph.D.)
- G. Kuncham (Ph.D.)
- S. Lee (Ph.D.)
- B. Michalowicz (Ph.D.)
- J. Oswal (Ph.D.)
- T. Tran (Ph.D.)
- L. Xu (P.h.D.)
- S. Xu (Ph.D.)
- J. Yao (Ph.D.)
- S. Zhang (Ph.D.)
- S. Mohammad (M.S.)
- B. Lampe (B.S.)
- N. Klein (B.S.)

Current Research Specialist

- R. Motlagh

Current Software Engineers

- N. Shineman
- M. Lieber

Past Students

- A. Awan (Ph.D.)
- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- M. Bayatpour (Ph.D.)
- R. Biswas (M.S.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- S. Chakraborthy (Ph.D.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- C.-H. Chu (Ph.D.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- R. Gulhane (M.S.)
- J. Hashmi (Ph.D.)
- M. Han (M.S.)
- W. Huang (Ph.D.)
- A. Jain (Ph.D.)
- J. Jani (M.S.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- M. Kedia (M.S.)
- K. S. Khorassani (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- P. Kousha (Ph.D.)
- K. Kulkarni (M.S.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- M. Li (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- J. Queiser (M.S.)
- K. Raj (M.S.)
- R. Rajachandrasekar (Ph.D.)
- B. Ramesh (Ph.D.)
- D. Shankar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- N. Sarkauskas (B.S. and M.S)
- V. Sathu (M.S.)
- N. Senthil Kumar (M.S.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Srivastava (M.S.)
- H. Subramoni (Ph.D.)
- S. Sur (Ph.D.)
- K. K. Suresh (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)
- J. Zhang (Ph.D.)
- Q. Zhou (Ph.D.)
- N. Chmura (B.S.)

Past Post-Docs

- D. Banerjee
- X. Besseron
- M. S. Ghazimirsaeed
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- K. Manian
- S. Marcarelli
- A. Ruhela
- J. Vienne
- H. Wang

Past Research Scientists

- K. Hamidouche
- S. Sur
- X. Lu
- M. Abduljabbar
- A. Shafi

Past Faculty

- H. Subramoni

Past Senior Research Associate

- J. Hashmi

Past Programmers

- A. Reifsteck
- D. Bureddy
- J. Perkins
- B. Seeds
- A. Guptha
- N. Pavuk

Past Research Specialist

- M. Arnold
- J. Smith

Funding Acknowledgments

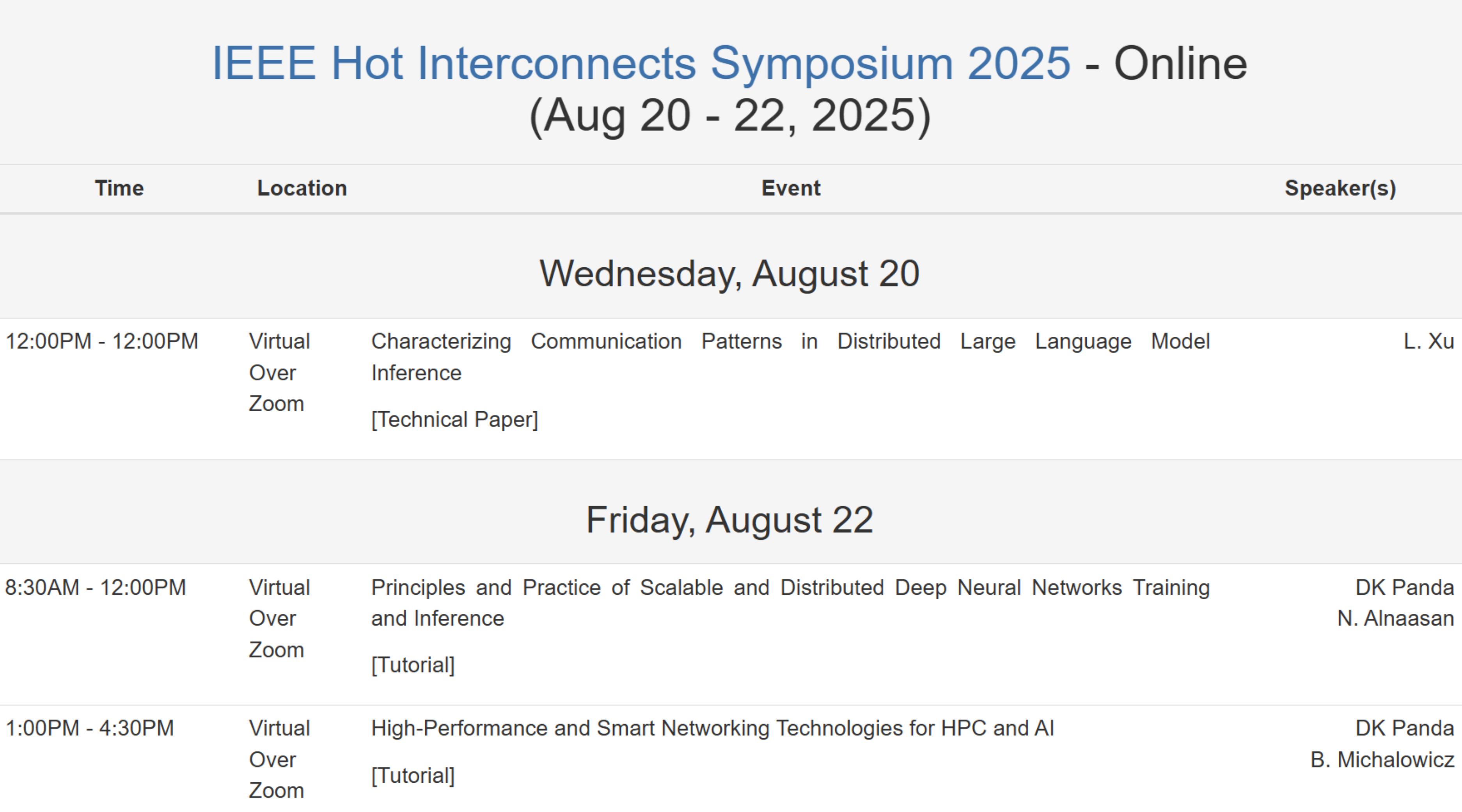
Funding Support by

Office of Science U.S. DEPARTMENT OF ENERGY
NSF
Mellanox TECHNOLOGIES
Microsoft
QLOGIC
arm
CRAY THE SUPERCOMPUTER COMPANY
BROADCOM
intel
LINUX NETWORX
NVIDIA
AMD
NetApp
Sun microsystems
CISCO
ORACLE
ROCKPORT
ETRI
CORNELIS NETWORKS

Equipment Support by

Mellanox TECHNOLOGIES
intel
arm
Sun microsystems
APPRO
NVIDIA
AMD
QLOGIC
advanced clustering technologies, inc.
Microway Technology you can count on
ROCKPORT
BROADCOM

Interested in more? Join us at IEEE Hot Interconnects 2025!



IEEE Hot Interconnects Symposium 2025 - Online (Aug 20 - 22, 2025)

Time	Location	Event	Speaker(s)
Wednesday, August 20			
12:00PM - 12:00PM	Virtual Over Zoom	Characterizing Communication Patterns in Distributed Large Language Model Inference [Technical Paper]	L. Xu
Friday, August 22			
8:30AM - 12:00PM	Virtual Over Zoom	Principles and Practice of Scalable and Distributed Deep Neural Networks Training and Inference [Tutorial]	DK Panda N. Alnaasan
1:00PM - 4:30PM	Virtual Over Zoom	High-Performance and Smart Networking Technologies for HPC and AI [Tutorial]	DK Panda B. Michalowicz

Hot Interconnects Registration (free) https://hoti.org/register.html

Thank You!

{alnaasan.1, yao.877}@osu.edu

Network Based Computing Laboratory

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

HiDL
High-Performance Deep Learning

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/