A Hybrid Tensor-Expert-Data Parallelism Approach to Optimize Mixture-of-Experts Training

Siddharth Singh¹, Olatunji Ruwase², Ammar Ahmad Awan², Samyam Rajbhandari², Yuxiong He², Abhinav Bhatelé¹

University of Maryland¹, Microsoft, Inc.²
Neural Networks

- Neural Networks (NNs): ‘Parameterized’ function approximators
- Can work with very high dimensional data.

Input ($x$) \quad \xrightarrow{\text{Organized into Layers}} \quad \text{Parameters ($\theta$)} \quad \xrightarrow{\text{Layer 0}} \quad \text{Layer 1} \quad \xrightarrow{\text{Output}} \quad f(x; \theta)

"A sitting dog"
Stochastic Gradient Descent

- Repeat until loss (L) has been minimized sufficiently:
  - Read in a batch of training data
  - Forward Pass: Calculate output $f(x; \theta)$ and the loss (L) on the batch.
  - Backward Pass: Calculate gradients of the loss wrt the parameters $\frac{\partial L}{\partial \theta}$.
  - Optimizer Step: Use $\frac{\partial L}{\partial \theta}$ to update $\theta$. 
Motivation

• More parameters = More accurate neural networks 😊

• However, more parameters = More FLOPs in training 😞

MoEs can make a given model arbitrarily large without changing its training FLOPs!
Mixture-of-Experts (MoEs)

Step 1: Start with a base model – usually a transformer neural network.

A Transformer Layer (Base Model)
Mixture-of-Experts (MoEs)

Step 2: Introduce multiple FF blocks. Each FF block is an ‘expert’.

A Transformer Layer (Base Model) + 2 experts
Mixture-of-Experts (MoEs)

Step 3: Introduce a parameterized router that maps tokens to experts.

A Transformer Layer (Base Model) + 2 experts
Mixture

Total FLOPs per token is independent of the number of experts!

Step 3: Introduce a parameterized router that maps tokens to experts.

A Transformer Layer (Base Model) + 2 experts
Caveat

- Diminishing returns beyond 64-128 experts.
- Imperative to increase base model sizes along with expert counts.

Figure courtesy Fedus et al. [2]
Gaps in Current Work

• Large MoEs are almost always trained in parallel on multiple GPUs.
• However, current parallel frameworks are not suited for MoEs built with large base models.
• They either support base models of limited sizes due to limited dimensions of parallelism.
  • Example – DeepSpeed-MoE [4] which has expert+data parallelism but not model parallelism (tensor/pipeline)
• Or, they use extremely inefficient parallel techniques like out-of-core training or FSDP.
  • Inefficiency occurs due to high communication times
Our work – Deepspeed-TED

- Goal 1 - Support MoEs with large base models
- Goal 2 - Minimize communication times to maintain efficiency.
- A three-dimensional hybrid of state-of-the-art parallel training algorithms
  - T – Tensor Parallelism (Megatron-LM [3])
  - E – Expert Parallelism (DeepSpeed-MoE [4])
  - D – Sharded Data Parallelism (ZeRO [5])
Data Parallelism

GPU - 0

GPU - 1

Average Gradients

Attention → Router

Replicate model
Partition Data
Average Gradients
Data + Expert Parallelism

GPU - 0

w0 \rightarrow \text{Attention} \rightarrow \text{Router} \rightarrow \text{FF0} \rightarrow \text{Router} \rightarrow \text{FF1} \rightarrow f0 \rightarrow f0

w1 \rightarrow \text{Attention} \rightarrow \text{Router} \rightarrow \text{FF0} \rightarrow \text{Router} \rightarrow \text{FF1} \rightarrow f1 \rightarrow f1

w2

w3

GPU - 1

w2 \rightarrow \text{Attention} \rightarrow \text{Router} \rightarrow \text{FF0} \rightarrow \text{Router} \rightarrow \text{FF1} \rightarrow f2 \rightarrow f2

w3 \rightarrow \text{Attention} \rightarrow \text{Router} \rightarrow \text{FF0} \rightarrow \text{Router} \rightarrow \text{FF1} \rightarrow f3 \rightarrow f3

All-to-All Communication

Divide Experts among GPUs
Tensor Parallelism

- Parallelize the matrix multiplications inside Attention and FF.
- Let’s first look at tensor parallelism in isolation.

Half the parameters!

GPU - 0

w0, w1, w2, w3

Attention

GPU - 1

w0, w1, w2, w3

Attention

TP 0

TP 1

Replicate Data
Tensor Parallelism

- Parallelize the matrix multiplications inside Attention and FF.
- Let’s first look at tensor parallelism in isolation.

![Diagram showing tensor parallelism with GPUs and attention layers](image)

GPU - 0

w0
w1
w2
w3

Attention

TP 0

a0'
a1'
a2'
a3'

GPU - 1

w0
w1
w2
w3

Attention

TP 1

a0'
a1'
a2'
a3'

Partial Outputs

a0
a1
a2
a3

All-Reduce Communication
Data + Expert + Tensor Parallelism

- Now let us look at tensor+expert+data parallelism on 8 GPUs.
- Two virtual topologies for Attention and FF-Blocks.
- For brevity, we will only look at GPUs [0-3]
TED for Attention (Non-Experts)

Attention
GPU - 1

Attention TP 0

Attention
GPU - 3

Attention TP 0

Attention TP 1

Distribution of Parameters

Data
Tensor
0 1 2 3
TED for Attention (Non-Experts)

GPU - 0

GPU - 1

GPU - 2

GPU - 3

Data

Tensor

0 1 2 3

Distribution of Input

Attention TP 0

Attention TP 1

Attention TP 0

Attention TP 1
TED for FF (Experts)

Expert 0
- GPU - 0
  - FF 0
  - TP 0

Expert 1
- GPU - 1
  - FF 1
  - TP 1

Expert 2
- GPU - 2
  - FF 1
  - TP 0

Expert 3
- GPU - 3
  - FF 1
  - TP 1
+ TED for FF (Experts)

GPU - 0

GPU - 1

GPU - 2

GPU - 3

All-To-All

All-Reduce

All-To-All

Expert

Tensor
3D parallelism helps up train larger models

Largest Trainable MoE Models on Summit

- Limit Number of experts to 128
- Limit tensor parallelism to a node.
Opt #1 Duplicate Token Dropping (DTD)

- Consider the first all-to-all.
- Tensor parallel GPUs communicate duplicate tokens.
- Remove this duplication to decrease All-To-All message sizes.
Opt #1 Duplicate Token Dropping

2x reduction in All-to-All volume

Cheap because intra-node

All-Gather
Opt #2 Communication-Aware Checkpointing (CAC)

- Reduces number of all-to-all and all-reduce calls by 33 percent by utilizing marginally extra memory.
- More details in paper.
Results

Batch Time Profile of a 6.7B base model + 16 experts on 128 GPUs of Summit

Nearly 50% time in communication!!
Results

Batch Time Profile of a 6.7B base model + 16 experts on 128 GPUs of Summit

Overall 21% Speedup
Results (Strong Scaling)

Strong Scaling of a 6.7B Base Model with Varying # Experts

Machine - Summit

22-29% speedups!
Results (Strong Scaling)

Strong Scaling for a 6.7B Base Model with 4 Experts

Machine - Summit
Conclusion and Future Work

• Developed DeepSpeed-TED, a highly scalable parallel framework for training high quality MoEs with large base models.

• Presented a three-dimensional hybrid parallel method that supports MoEs with 4-8x larger models than the SoTA.

• Introduced communication optimizations that can achieve significant reductions in the collective communication times.

• As future work, we want to explore pipeline parallelism as a fourth dimension to scale to even larger base models.
Our work is integrated in DeepSpeed, a widely used open-source framework for parallel deep learning.

URL - https://github.com/microsoft/DeepSpeed
Bibliography


