MEGATRON-LM

TRAINING MULTI-BILLION PARAMETER LANGUAGE MODELS USING MODEL PARALLELISM

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, Bryan Catanzaro
Large Models Require Large Memory

- Training >> Inference
- Adam > SGD
- Larger minibatch
- More parameters
- ...
- More devices are needed to:
  - Speed up training
  - Simply enable training

![GPU VRAM Usage Estimation for training different models]
Data Parallelism

• Partitions a training minibatch across multiple devices
• Linearly scalable
• Slicing activation only
• Does not help with excessive model size
Pipeline Parallelism

• Layer-wise model parallelism
• Bubbles reduce efficiency
• Requires additional logic to handle pipelining

![Pipeline schedule of Gpipe](https://developer.nvidia.com/blog/scaling-language-model-training-to-a-trillion-parameters-using-megatron/)

Pipeline Parallelism

- Bubbles reduce efficiency
  - Larger batch sizes for relatively small bubbles are impractical
  - Low scaling efficiency

\[
\text{bubble time fraction} = \frac{p - 1}{m}
\]


Tensor Parallelism

• Intra-layer model-parallelism
• Good scaling efficiency inside one node
• Orthogonal to DP and PP: combine to get best strategy

512 GPUs: 15.1 PFLOPs, 76% scaling efficiency
Model + Data Parallel: 64-way DP
Model Parallel: 1 – 8 GPUs for 1B – 8B models
Baseline: 0.039 PFLOPs (30% of peak FLOPs)
Tensor Parallelism

• Tailored for transformer networks

• Transformer Layer
  • Self-attention block
  • Two-layer MLP

• Targets:
  • Exploit parallelism wherever possible
  • Reduce synchronization cost
Tensor Parallelism: MLP

• Two layers: $Y = \text{GeLU}(XA)$, $Z = \text{Dropout}(YB)$

• Option 1: Split $A$ and $B$ along their rows
  • $X = [X_1, X_2], A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$
  • $Y = \text{GeLU}(X_1A_1 + X_2A_2) \neq \text{GeLU}(X_1A_1) + \text{GeLU}(X_2A_2)$
  • Synchronizations before GeLU and Dropout

\[ f: \text{SPLIT} \text{ (forward)} \]
\[ \text{ALL}_G\text{ATHER} \text{ (backward)} \]
\[ g: \text{ALL}_R\text{EDUCE} \text{ (forward)} \]
\[ \text{IDENTITY} \text{ (backward)} \]
Tensor Parallelism: MLP

- Two layers: $Y = \text{GeLU}(XA)$, $Z = \text{Dropout}(YB)$
- Option 2: Split $A$ along its columns
  - $A = [A_1, A_2]$
  - $Y = [Y_1, Y_2] = \text{GeLU}([XA_1, XA_2]) = [\text{GeLU}(XA_1), \text{GeLU}(XA_2)]$
- Split $B$ along its rows
- Only one sync needed before Dropout

- $f$: IDENTITY (forward)
  ALL_REDUCE (backward)
- $g$: ALL_REDUCE (forward)
  IDENTITY (backward)
Tensor Parallelism: Self-Attention

- Attention block: Self-attention followed by a linear layer
  - Partition K, Q, V along their columns
  - Slice the subsequent MM weight along its rows
  - Only one sync required
Tensor Parallelism: Communication cost

- 4 Total communication operations in 1 forward + backward pass
Tensor Parallelism: I/O embedding

- Embedding matrix $E_{H \times v}$: Hidden-size $\times$ vocab-size
  - Weights shared between input and output embeddings
- Parallelized along columns (vocab dimension)
- Input embedding: acquired using ALL_REDUCE

$$E = \begin{pmatrix} E_1 \\ E_2 \end{pmatrix}$$

$$H \cdot E = \begin{pmatrix} v/2 \\ v/2 \end{pmatrix}$$

$f$: SPLIT (forward)
ALL_GATHER (backward)

$g$: ALL_REDUCE (forward)
IDENTITY (backward)
Tensor Parallelism: I/O embedding

- Output embedding: acquired using ALL_GATHER
- Costly: This will communicate $b \times s \times v$ elements
  - $b$: batch size
  - $s$: sequence length

\[
\begin{align*}
H \cdot E &= \begin{pmatrix} v/2 & v/2 \\ E_1 & E_2 \end{pmatrix} \\
Y &= \text{ALL_GATHER} \left( \begin{pmatrix} v/2 & v/2 \\ XE_1 & XE_2 \end{pmatrix} \right)
\end{align*}
\]

$f$: IDENTITY (forward)
ALL_REDUCE (backward)
$g$: ALL_GATHER (forward)
SPLIT (backward)
Tensor Parallelism: I/O embedding

- Output embedding: acquired using ALL_GATHER
  - Fuse the output \([Y_1, Y_2]\) with the cross-entropy loss
  - Communication reduced to \(b \times s\)
Experiments

- 32 DGX-2H servers
  - 512 V100 SXM3 32GB GPUs
- Intra-server connection
  - 300 GB/s NVSwitch
- Inter-server connection
  - 100 GB/s InfiniBand (8 per server)

- GPT-2 & BERT Models
- TP (+ DP)
- Mixture of datasets
Experiments: Scalability

- GPT-2: 1B – 8B
- Hidden size: 96
- Parameters/GPU: ~1B
- Weak Scaling @ 512 GPUs: 74%
Experiments: GPT-2

- 0.4B, 2.5B, 8.3B

<table>
<thead>
<tr>
<th>Parameter Count</th>
<th>Layers</th>
<th>Hidden Size</th>
<th>Attn Heads</th>
<th>Hidden Size per Head</th>
<th>Total GPUs</th>
<th>Time per Epoch (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>355M</td>
<td>24</td>
<td>1024</td>
<td>16</td>
<td>64</td>
<td>64</td>
<td>0.86</td>
</tr>
<tr>
<td>2.5B</td>
<td>54</td>
<td>1920</td>
<td>20</td>
<td>96</td>
<td>128</td>
<td>2.27</td>
</tr>
<tr>
<td>8.3B</td>
<td>72</td>
<td>3072</td>
<td>24</td>
<td>128</td>
<td>512</td>
<td>2.10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>LAMBADA Perplexity ↓</th>
<th>LAMBADA Accuracy ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>355M</td>
<td>19.31</td>
<td>45.18%</td>
</tr>
<tr>
<td>2.5B</td>
<td>12.76</td>
<td>61.73%</td>
</tr>
<tr>
<td>8.3B</td>
<td><strong>10.81</strong></td>
<td><strong>66.51%</strong></td>
</tr>
<tr>
<td>Previous SOTA</td>
<td>15.79</td>
<td>63.24%</td>
</tr>
</tbody>
</table>
Experiments: BERT

- 0.3B, 1.3B, 3.9B
- Modified architecture from (a) to (b) to allow for larger models
Combining Multiple Parallelism Techniques

- Model_Parallelism = Tensor_Parallelism × Pipeline_Parallelism
Combining Multiple Parallelism Techniques

- Best practice can be chosen from multiple combinations


Megatron-LM: Training Multi-billion Parameter Language Models Using Model Parallelism
Combining Multiple Parallelism Techniques

• Further reduce communication cost between parallel layers

Combining Multiple Parallelism Techniques

Figure 10. Throughput per GPU with and without the scatter/gather optimization for a GPT model with 175 billion parameters using 96 A100 GPUs and the interleaved schedule.


Megatron-LM: Training Multi-billion Parameter Language Models Using Model Parallelism
Combining Multiple Parallelism Techniques

\[(\text{MP} = \text{TP} \times \text{PP}) \text{ GPUs} = 1 \text{ Virtual GPU}\]
Combining Multiple Parallelism Techniques

• $\text{DP\_max} = \frac{\text{(# of GPUs)}}{\text{(# of Virtual GPUs)}}$

Discussion

• Efficient model parallelism with good scalability
• Compatible with other parallelism methods
• Very useful codebase (https://github.com/NVIDIA/Megatron-LM)

• TP brings large communication cost
  • The limit of efficient TP depends largely on # of GPUs on a single node
  • Speed will be significantly affected if intra-node connection is bad
  • One slow GPU will make the entire system much slower
  • Dependencies prevent interleaving of communication and computation[1]

THANK YOU!

Presented by Yufeng Du