Enabling efficient trillion parameter scale training for deep learning models

https://github.com/microsoft/DeepSpeed

Presented by: Olatunji (Tunji) Ruwase
On behalf of the DeepSpeed team
Motivation: Why large language models?

- Larger models $\rightarrow$ better accuracy
- Model size is still growing
- Not reached the accuracy limit yet
- More compute-efficient to train larger models than smaller ones to same accuracy
System **Challenges/Opportunities** of Large language models?

- **Memory**
- **Compute**
- **Data**
ZeRO, ZeRO-Offload, ZeRO-Infinity

Breaking the GPU Memory Wall for DL Training
Understanding Memory Consumption

- FP(BF)16 parameter: 2M bytes
- FP(BF)16 Gradients: 2M bytes
- FP32 Optimizer States: 16M bytes
  - Gradients, Variance, Momentum, Parameters

Example 1B parameter model -> 20GB/GPU

Memory consumption doesn’t include:
- Input batch + activations

\[ M = \text{number of parameters in the model} \]

*Mixed Precision Training* (ICLR ‘18) with Adam Optimizer
ZeRO: Overcoming GPU memory wall

- Family of composable optimizations to reduce GPU memory costs of DL state (params, grads, optimizer)
- **Partitioning** DL state across data parallel GPUs (3 stages)
- **Offloading** DL state to CPU or NVMe memories
ZeRO: Overcoming GPU memory wall

- Family of composable optimizations to reduce GPU memory costs of DL state (params, grads, optimizer)
- **Partitioning** DL state across data parallel GPUs (3 stages)
- **Offloading** DL state to CPU or NVMe memories
ZeRO: Overcoming GPU memory wall

• Family of composable optimizations to reduce GPU memory costs of DL state (params, grads, optimizer)

• **Partitioning** DL state across data parallel GPUs (3 stages)

• **Offloading** DL state to CPU or NVMe memories

![Diagram showing Stage 1 (P_{os})](image-url)
ZeRO: Overcoming GPU memory wall

- Family of composable optimizations to reduce GPU memory costs of DL state (params, grads, optimizer)
- **Partitioning** DL state across data parallel GPUs (3 stages)
- **Offloading** DL state to CPU or NVMe memories

![Diagram showing baseline and two partition stages](image)
ZeRO: Overcoming GPU memory wall

- Family of composable optimizations to reduce GPU memory costs of DL state (params, grads, optimizer)
- **Partitioning** DL state across data parallel GPUs (3 stages)
- **Offloading** DL state to CPU or NVMe memories
ZeRO: Overcoming GPU memory wall

- Family of composable optimizations to reduce GPU memory costs of DL state (params, grads, optimizer)
- **Partitioning** DL state across data parallel GPUs (3 stages)
- **Offloading** DL state to CPU or NVMe memories

<table>
<thead>
<tr>
<th>Bytes/param/GPU</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 + 2 + 16 = 20</td>
<td>(2 + 2 + \frac{16}{N}) &lt; 5</td>
</tr>
<tr>
<td>2 + 2 + (16/N) &lt; 5</td>
<td>(2 + \frac{(2+16)}{N}) &lt; 3</td>
</tr>
<tr>
<td>(2 + 2 + 16)/N &lt; 1</td>
<td>(\frac{(2 + 2 + 16)}{N}) &lt; 1</td>
</tr>
</tbody>
</table>
ZeRO: Overcoming GPU memory wall

- Family of composable optimizations to reduce GPU memory costs of DL state (params, grads, optimizer)
- **Partitioning** DL state across data parallel GPUs (3 stages)
- **Offloading** DL state to CPU or NVMe memories

![Diagram showing ZeRO-Offload]
ZeRO: Overcoming GPU memory wall

- Family of composable optimizations to reduce GPU memory costs of DL state (params, grads, optimizer)
- **Partitioning** DL state across data parallel GPUs (3 stages)
- **Offloading** DL state to CPU or NVMe memories

---

Offload

ZeRO-Infinity

Baseline

$\text{gpu}_0 \quad \ldots \quad \text{gpu}_i \quad \ldots \quad \text{gpu}_{N-1}$

$P_{os}$

$P_{os-g}$

$P_{os+g+p}$

- Parameters
- Gradients
- Optimizer States

• Offloading DL state to CPU or NVMe memories
ZeRO-Infinity

Breaking the GPU Memory Wall for DL Training
Large model training landscape

- GPU Memory Wall
  - 1T (10T) params: 800 (8K) V100 GPUs
  - How do we support the growth in model size?

- Accessibility to large model training
  - 256 GPUs to fine-tune GPT-3
  - Limited access to such resources

- Model code refactoring
  - Re-writing the model using 3D parallelism (tensor-slicing + pipeline parallelism)
  - Painful and error prone

*AI and Memory Wall. (This blogpost has been written in... | by Amir Gholami | riselab | Medium*
Beyond the GPU Memory

- Modern clusters have heterogeneous memory systems.

- GPU memory is a small fraction

- Leverages GPU/CPU/NVMe memory
  - 32T params on 32 nodes
  - 1T params on a single node

- Fine-tune GPT-3 size on single node
Recap: Large model training landscape today

- **GPU Memory Wall**
  - 1T (10T) params: 800 (8K) V100 GPUs
  - How do we support the growth in model size?

- **Accessibility to large model training**
  - 256 GPUs to fine-tune GPT-3
  - Limited access to such resources

- **Model code refactoring**
  - Re-writing the model using 3D parallelism (tensor-slicing + pipeline parallelism)
  - Painful and error prone
Redefining the landscape with ZeRO-Infinity

- Beyond GPU Memory
  - 50x larger models
  - 32T params on 512 GPUs (instead of 25K)

- Broader access to large model traini
  - GPT-3 sized fine-tuning on a single node/GPU (instead of 16 nodes)

- Excellent Throughput and Scalability
  - Comparable to 3D-parallelism

- Ease of Use
  - No model refactoring necessary

Paper: ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning (arxiv.org)
DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale

Sanyam Rajbhandari, Conglong Li, Zhenwei You, Mingia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Jwan, Jeff Rasley, Yuexue He

Abstract
As the training of giant dense models hits the boundary on the availability and capability of the hardware resources today, mixture-of-experts (MoE) models have become one of the most promising model architectures due to their significant training cost reduction compared to a single model. Their training cost saving is demonstrated from encoder-decoder models (prior work) to a 5x saving for auto-aggressive language models (this work). However, due to the much larger model size and unique architecture, how to provide fast MoE model inference remains challenging and unsolved, limiting their practical usage. To tackle this, we present DeepSpeed-MoE, an end-to-end MoE training system.

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

Siddharth Singh
Department of Computer Science,
University of Maryland, College Park, Maryland, USA

Olahanjie Bawuse
olahanjie@microsoft.com
Microsoft, Inc.
Redmond, Washington, USA

Ammar Ahmad Jwan
ammar.jwan@microsoft.com
Microsoft, Inc.
Redmond, Washington, USA

Sanyam Rajbhandari
sanyam@microsoft.com
Microsoft, Inc.
Redmond, Washington, USA

Yuexue He
yuxuehe@microsoft.com
Microsoft, Inc.
Redmond, Washington, USA

Abyzua Bhatti
bhatti@cs.umd.edu
Department of Computer Science,
University of Maryland, College Park, Maryland, USA

ABSTRACT
Mixture-of-Experts (MoE) is a neural network architecture that adds expert-oriented expert blocks to a base model, increasing the number of parameters without impacting computational costs.

1 INTRODUCTION
Contemporary state-of-the-art AI algorithms have come to rely on neural networks such as GPT-3 (4) and EF-T5 (14) with hundreds of billions of parameters. However, training or running inference
Mixture of Experts (MoE): Overview

- MoE models have been around for a while.

- Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer
  - Harder to scale, instability during training, and inefficient training

- GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding
  - 600B models beating 96-layer dense models, 10x training speedup, generic sharding framework (Tensorflow XLA), full precision training
  - Less stability with larger models

- Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
  - More efficient training
    - Top-1 gating instead of top-2/top-k, Better initialization conditions, Mixed precision training: FP32 gating (instead of FP16), Stable training with larger models
  - SOTA results on language understanding task
MoE: Gshard and Switch Transformer

Figure 3: Illustration of scaling of Transformer Encoder with MoE Layers. The MoE layer replaces the every other Transformer feed-forward layer. Decoder modification is similar. (a) The encoder of a standard Transformer model is a stack of self-attention and feed forward layers interleaved with residual connections and layer normalization. (b) By replacing every other feed forward layer with a MoE layer, we get the model structure of the MoE Transformer Encoder. (c) When scaling to multiple devices, the MoE layer is sharded across devices, while all other layers are replicated.

Figure 2: Illustration of a Switch Transformer encoder block. We replace the dense feed forward network (FFN) layer present in the Transformer with a sparse Switch FFN layer (light blue). The layer operates independently on the tokens in the sequence. We diagram two tokens ($x_1 = \text{“More”}$ and $x_2 = \text{“Parameters”}$ below) being routed (solid lines) across four FFN experts, where the router independently routes each token. The switch FFN layer returns the output of the selected FFN multiplied by the router gate value (dotted-line).
MoE models are sparse and need less compute

**Dense Models:**
- All parameters are used in forward and backward paths
- Increasing model capacity needs more computation
- Optimized for dense computation
- Larger model size $\rightarrow$ Higher compute requirements (FLOPs)

**Sparse MoE models**
- Sparse utilization of subset of parameters based on input
- Same computation is needed regardless of the model size
- Not-optimized for dense computation
- Larger model size $\rightarrow$ Similar/Same Compute requirements

Figure 2: Illustration of a Switch Transformer encoder block. We replace the dense feed-forward network (FFN) layer present in the Transformer with a sparse Switch FFN layer (light blue). The layer operates independently on the tokens in the sequence. We diagram two tokens ($x_1$ = “More” and $x_2$ = “Parameters” below) being routed (solid lines) across four FFN experts, where the router independently routes each token. The switch FFN layer returns the output of the selected FFN multiplied by the router gate value (dotted line).

Figure 3: Illustration of scaling of Transformer Encoder with MoE Layers. The MoE layer replaces every other Transformer feed-forward layer. Decoder modification is similar. (a) The encoder of a standard Transformer model is a stack of self-attention and feed-forward layers interleaved with residual connections and layer normalization. (b) By replacing every other feed-forward layer with a MoE layer, we get the model structure of the MoE Transformer Encoder. (c) When scaling to multiple devices, the MoE layer is shared across devices, while all other layers are replicated.
What is Expert Parallelism?

- **Expert Parallelism --> Data and Model parallelism within the model**
  - Non-MoE parameters – replicated
    - Like standard data parallelism (DP)
    - ZeRO DP in DeepSpeed can shard these too!
  - MoE parameters – partitioned (sharded)
    - Like model parallelism (MP)
  - Two All-to-All(s) in Forward and Backward
Next AI Scale on current hardware

• Can we achieve next generation model quality on current generation of hardware?

• From a training perspective MoE provides a promising path
  • Scale at sub-linear cost
  • Z-Code multi-lingual multi-task model

• MoE is *promising* but is it *practical*?
  • **Limited Scope**: Does it work for NLG or NLR or other models?
  • **Massive Memory Requirements**: 8-10x in size compared to quality equivalent dense
  • **Limited expert scaling**: Diminishing returns at 64-128 experts?
DeepSpeed MoE: Multidimensional Parallelism

<table>
<thead>
<tr>
<th>Short Name</th>
<th>Flexible Parallelism Combinations</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Expert</td>
<td>Scales the model size by increasing the number of experts</td>
</tr>
<tr>
<td>E+D</td>
<td>Expert + Data</td>
<td>Accelerates training throughput by scaling to multiple data parallel groups</td>
</tr>
<tr>
<td>E+Z</td>
<td>Expert + ZeRO</td>
<td>Partitions the nonexpert parameters to support larger base models</td>
</tr>
<tr>
<td>E+D+M</td>
<td>Expert + Data + Model</td>
<td>Supports massive hidden sizes and even larger base models than E+Z</td>
</tr>
<tr>
<td>E+D+Z</td>
<td>Expert + Data + ZeRO</td>
<td></td>
</tr>
<tr>
<td>E+Z-Off+M</td>
<td>Expert + ZeRO-Offload + Model</td>
<td>Leverages both GPU and CPU memory for large MoE models on limited GPU resources</td>
</tr>
</tbody>
</table>

Can scale both: 1) Number of experts and 2) Base model sizes
Cheaper NLG Model Training with MoE

- 1.3B+MoE with 128 experts, compared to 1.3B and 6.7B dense (GPT-3 like)
- **5x** lower training cost to same accuracy using MoE
- **8x** more parameters to same accuracy using MoE

<table>
<thead>
<tr>
<th>Case</th>
<th>Model size</th>
<th>LAMBADA: completion prediction</th>
<th>PIQA: commonsense reasoning</th>
<th>BoolQ: reading comprehension</th>
<th>RACE-h: reading comprehension</th>
<th>TriviaQA: question answering</th>
<th>WebQs: question answering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense NLG:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) 350M</td>
<td>350M</td>
<td>52.03</td>
<td>69.31</td>
<td>53.64</td>
<td>31.77</td>
<td>3.21</td>
<td>1.57</td>
</tr>
<tr>
<td>(2) 1.3B</td>
<td>1.3B</td>
<td>63.65</td>
<td>73.39</td>
<td>63.39</td>
<td>35.60</td>
<td>10.05</td>
<td>3.25</td>
</tr>
<tr>
<td>(3) 6.7B</td>
<td>6.7B</td>
<td>71.94</td>
<td><strong>76.71</strong></td>
<td><strong>67.03</strong></td>
<td>37.42</td>
<td>23.47</td>
<td>5.12</td>
</tr>
<tr>
<td>Standard MoE NLG:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4) 350M+MoE-128</td>
<td>13B</td>
<td>62.70</td>
<td>74.59</td>
<td>60.46</td>
<td>35.60</td>
<td>16.58</td>
<td>5.17</td>
</tr>
<tr>
<td>(5) 1.3B+MoE-128</td>
<td>52B</td>
<td>69.84</td>
<td><strong>76.71</strong></td>
<td>64.92</td>
<td><strong>38.09</strong></td>
<td><strong>31.29</strong></td>
<td><strong>7.19</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Training samples per sec</th>
<th>Throughput gain/ Cost Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.7B dense</td>
<td>70</td>
</tr>
<tr>
<td>1.3B+MoE-128</td>
<td>372</td>
</tr>
</tbody>
</table>
Parameter Efficient MoE via PR-MoE and MoS

- Challenges of MoE: 8x parameters than quality equivalent dense models
  - Training requires large memory footprint
  - Slow inference due to parameter loading

- New opportunities
  - Parameter efficient MoE
    - Homogeneous layer structure → Pyramid MoE
Parameter Efficient MoE via PR-MoE and MoS

- Challenges of MoE: 8x parameters than quality equivalent dense models
  - Training requires large memory footprint
  - Slow inference due to parameter loading

- New opportunities
  - Parameter efficient MoE
    - Homogeneous layer structure $\rightarrow$ Pyramid MoE
    - Plain structure $\rightarrow$ Residual MoE

Top-2 MoE with extra communication
Parameter Efficient MoE via PR-MoE and MoS

- Challenges of MoE: 8x parameters than quality equivalent dense models
  - Training requires large memory footprint
  - Slow inference due to parameter loading

- New opportunities
  - Parameter efficient MoE
    - Homogeneous layer structure → Pyramid MoE
    - Plain structure → Residual MoE
Parameter Efficient MoE via PR-MoE and MoS

- Challenges of MoE: 8x parameters than quality equivalent dense models
  - Training requires large memory footprint
  - Slow inference due to parameter loading

- New opportunities
  - Parameter efficient MoE
    - Homogeneous layer structure → Pyramid MoE
    - Plain structure → Residual MoE
    - Our design PR-MoE
Parameter Efficient MoE via PR-MoE and MoS

- Challenges of MoE: 8x parameters than quality equivalent dense models
  - Training requires large memory footprint
  - Slow inference due to parameter loading

- New opportunities
  - Parameter efficient MoE
    - Homogeneous layer structure → Pyramid structure
    - Plain structure → Residual structure
    - Our design PR-MoE
  - MoS: MoE-to-MoE Knowledge distillation
**Parameter Efficient MoE via PR-MoE and MoS**

- PR-MoE: model size reduction from **1.7x** to **3.2x**
- PR-MoE + MoS: model size reduction from **1.9x** to **3.7x**

<table>
<thead>
<tr>
<th>Case</th>
<th>Model size (Reduction)</th>
<th>LAMBADA</th>
<th>PIQA</th>
<th>BoolQ</th>
<th>RACE-h</th>
<th>TriviaQA</th>
<th>WebQs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoE NLG with 350M base model:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) MoE</td>
<td>13B (1x)</td>
<td>62.70</td>
<td>74.59</td>
<td>60.46</td>
<td>35.60</td>
<td>16.58</td>
<td>5.17</td>
</tr>
<tr>
<td>(2) PR-MoE</td>
<td>4.0B (3.2x)</td>
<td><strong>63.65</strong></td>
<td>73.99</td>
<td>59.88</td>
<td><strong>35.69</strong></td>
<td>16.30</td>
<td>4.73</td>
</tr>
<tr>
<td>(3) PR-MoE + MoS</td>
<td><strong>3.5B (3.7x)</strong></td>
<td>63.46</td>
<td>73.34</td>
<td>58.07</td>
<td>34.83</td>
<td>13.69</td>
<td><strong>5.22</strong></td>
</tr>
<tr>
<td>MoE NLG with 1.3B base model:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4) MoE</td>
<td>52B (1x)</td>
<td>69.84</td>
<td>76.71</td>
<td>64.92</td>
<td><strong>38.09</strong></td>
<td>31.29</td>
<td>7.19</td>
</tr>
<tr>
<td>(5) PR-MoE</td>
<td>31B (1.7x)</td>
<td><strong>70.60</strong></td>
<td><strong>77.75</strong></td>
<td><strong>67.16</strong></td>
<td><strong>38.09</strong></td>
<td>28.86</td>
<td>7.73</td>
</tr>
<tr>
<td>(6) PR-MoE + MoS</td>
<td><strong>27B (1.9x)</strong></td>
<td>70.17</td>
<td>77.69</td>
<td>65.66</td>
<td>36.94</td>
<td>29.05</td>
<td><strong>8.22</strong></td>
</tr>
</tbody>
</table>

**Paper:** [DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale (ICML'22)](https://example.com/deepspeed-moe)
Deepspeed-TED: Scaling MoE base model

• Enable MoEs with large base models
• 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])
3D parallelism helps up train larger models

- Limit Number of experts to 128
- Limit tensor parallelism to a node.

Largest Trainable MoE Models on Summit

4.8x increase!
Results

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

Nearly 50% time in communication!!
Results: Communication optimizations

Batch Time Profile of a 6.7B base model + 16 experts on 128 GPUs of Summit
Results (Strong Scaling)

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

Machine - Summit

22-29% speedups!
Data Efficiency

Improving Deep Learning Model Quality and Training Efficiency via Efficient Data Sampling and Routing
DeepSpeed Data Efficiency

- Why we care about data efficiency
  - Training cost = $O(\text{model scale} \times \text{data scale})$
  - Data scale is increasing as fast as model scale

- Our goal
  - Achieve same model quality with less data
  - Achieve better model quality with same data
  - No/minimal model architecture change
The Stability-Efficiency Dilemma

• When pretraining large-scale language models
  • We want large batch sizes & learning rates to increase training efficiency
  • But they affect training stability, causing poor convergence or divergence
The Stability-Efficiency Dilemma

• When pretraining large-scale language models
  • We want large batch sizes & learning rates to increase training efficiency
  • But they affect training stability, causing poor convergence or divergence

• We study this dilemma by in-depth analysis on GPT-2 pretraining
  • Larger batch sizes & LR reduces training time, but affect model quality
  • (In paper) A correlation between training instability and gradient variance

<table>
<thead>
<tr>
<th>GPT-2 1.5B model</th>
<th>Training time</th>
<th>WikiText PPL ↓</th>
<th>LAMBADA Acc ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size 512</td>
<td>341Hr</td>
<td>13.89</td>
<td>57.29%</td>
</tr>
<tr>
<td>Batch size 4K, 4x LR</td>
<td>151Hr</td>
<td>14.76</td>
<td>55.06%</td>
</tr>
</tbody>
</table>
The Stability-Efficiency Dilemma

• When pretraining large-scale language models
  • We want large batch sizes & learning rates to increase **training efficiency**
  • But they affect **training stability**, causing poor convergence or divergence

• We study this dilemma by in-depth analysis on GPT-2 pretraining
  • Larger batch sizes & LR reduces training time, but affect model quality
  • (In paper) A correlation between training instability and gradient variance

<table>
<thead>
<tr>
<th>GPT-2 1.5B model</th>
<th>Training time</th>
<th>WikiText PPL ↓</th>
<th>LAMBADA Acc ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size 512</td>
<td>341Hr</td>
<td>13.89</td>
<td>57.29%</td>
</tr>
<tr>
<td>Batch size 4K, 4x LR</td>
<td>151Hr</td>
<td>14.76</td>
<td>55.06%</td>
</tr>
</tbody>
</table>
Proposed Sequence Length Warmup Method

• Instability mostly at early stage $\rightarrow$ some sort of “warmup” is needed
  • LR and batch size warmup didn’t help
Proposed Sequence Length Warmup Method

- Instability mostly at early stage → some sort of “warmup” is needed
  - LR and batch size warmup didn’t help
  - Our study shows warming up sequence length is promising
Proposed Sequence Length Warmup Method

• Instability mostly at early stage $\rightarrow$ some sort of “warmup” is needed
  • LR and batch size warmup didn’t help
  • Our study shows warming up sequence length is promising

• The Sequence Length Warmup (SLW) method
  • Two configs: starting sequence length, number of warmup steps
  • (In paper) Simple truncation-based implementation, low-cost tuning strategy
GPT-2 Evaluation

- Stable training under large batch size & LR
- Same model quality under less token/time

<table>
<thead>
<tr>
<th>GPT-2 1.5B model</th>
<th>Training time</th>
<th>Training tokens</th>
<th>WikiText PPL ↓</th>
<th>LAMBADA Acc ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size 512, baseline</td>
<td>341Hr</td>
<td>157B</td>
<td>13.89</td>
<td>57.29%</td>
</tr>
<tr>
<td>Batch size 4K, 4x LR, baseline</td>
<td>151Hr</td>
<td>157B</td>
<td>14.76</td>
<td>55.06%</td>
</tr>
<tr>
<td>Batch size 4K, 4x LR, ours</td>
<td>121Hr</td>
<td>121B</td>
<td>13.88</td>
<td>58.20%</td>
</tr>
</tbody>
</table>
GPT-2 Evaluation

- Stable training under large batch size & LR
- Same model quality under less token/time
- Even better model quality under same token/time

<table>
<thead>
<tr>
<th>GPT-2 1.5B model</th>
<th>Training time</th>
<th>Training tokens</th>
<th>WikiText PPL ↓</th>
<th>LAMBADA Acc ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size 512, baseline</td>
<td>341Hr</td>
<td>157B</td>
<td>13.89</td>
<td>57.29%</td>
</tr>
<tr>
<td>Batch size 4K, 4x LR, baseline</td>
<td>151Hr</td>
<td>157B</td>
<td>14.76</td>
<td>55.06%</td>
</tr>
<tr>
<td>Batch size 4K, 4x LR, ours</td>
<td>121Hr</td>
<td>121B</td>
<td>13.88</td>
<td>58.20%</td>
</tr>
<tr>
<td>Batch size 4K, 4x LR, ours</td>
<td>155Hr</td>
<td>157B</td>
<td>13.72</td>
<td>58.47%</td>
</tr>
</tbody>
</table>
DeepSpeed: Reshaping the Large Model Training Landscape

System capability to efficiently train models with **trillions of parameters**

*AI and Memory Wall. (This blogpost has been written in... | by Amir Gholami | riselab | Medium*
DeepSpeed Transformation

**Past**
Training optimization library

- Training
  - Speed
  - Scale
  - Cost
  - Democratization

**Today and Future**
Multi-purpose DL optimization suite

- Training
  - Speed
  - Scale
  - Cost
  - Democratization
  - MoE models
  - Long sequence
  - RLHF

- Inference
  - Large models
  - Latency
  - Serving cost
  - Agility

- Compression
  - Model size
  - Latency
  - Composability
  - Runnable on client devices

- Science
  - Speed
  - Scale
  - Capability
  - Diversity
We welcome contributions! Make your first pull request 😊

https://github.com/microsoft/DeepSpeed

www.deepspeed.ai

Follow us on X: @MSFTDeepSpeed

Thank You!
ZeRO-Infinity in Action