ZeRO: Memory Optimizations Toward Training Trillion Parameter Models

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, Yuxiong He

Presenter: Siyu Chen

03/02/2022

Background

- Number of DNN parameters
  - Past: million
    - AlexNet (60M), VGG-19 (144M), ResNet-50 (26M)

Background

- Number of DNN parameters
  - Past: million
    - AlexNet (60M), VGG-19 (144M), ResNet-50 (26M)
  - Present: billion
    - Megatron-LM (8.3B), T5 (11B), V-MoE-15B*, SwinV2-G (3B)

* indicates sparse models

Background

- Number of DNN parameters
  - Past: million
    - AlexNet (60M), VGG-19 (144M), ResNet-50 (26M)
  - Present: billion
    - Megatron-LM (8.3B), T5 (11B), V-MoE-15B*, SwinV2-G (3B)
  - Future: trillion?
    - GPT-3 (0.18T), Switch-C (1.6T)*, MT-NLG (0.53T)

* indicates sparse models

Background

- Number of DNN param
  - Past: million
    - AlexNet (60M)
  - Present: billion
    - Megatron-LM
  - Future: trillion?
    - GPT-3 (0.18T),

Figure from NVIDIA website

Background

- Number of DNN parameters
  - Past: million
    - AlexNet (60M), VGG-19 (144M), ResNet-50 (26M)
  - Present: billion
    - Megatron-LM (8.3B), T5 (11B), V-MoE-15B*, SwinV2-G (3B)
  - Future: trillion?
    - GPT-3 (0.18T), Switch-C (1.6T)*, MT-NLG (0.53T)
- GPU architecture
  - Past: V100, 32GB memory, 130TFLOPS
  - Present: A100, 80GB memory, 310TFLOPS

* indicates sparse models

Problem

- Large models cannot fit within current device memory
- Issues with previous parallelism
  - Data Parallelism (DP)
    - Good compute/communication efficiency
    - Poor memory efficiency

Problem

- Large models cannot fit within current device memory
- Issues with previous parallelism
  - Data Parallelism (DP)
    - Good compute/communication efficiency
    - Poor memory efficiency
  - Model Parallelism (MP)
    - Poor compute/communication efficiency
    - Good memory efficiency

Problem

- Large models cannot fit within current device memory
- Issues with previous parallelism
  - Data Parallelism (DP)
    - Good compute/communication efficiency
    - Poor memory efficiency
  - Model Parallelism (MP)
    - Poor compute/communication efficiency
    - Good memory efficiency
  - Pipeline Parallelism (PP)
    - G-Pipe: require large batch size to hide bubble
    - PipeDream: store multiple copies of stale parameters

Analysis: Memory Consumption

- Major memory usage: model states and activation checkpoints

Params (Trillions)	Layers	Hidden Size	Attn Heads	Model States (TB/Model)	TB/Node	
					Act.	Act. Ckpt.
0.10	80	10K	128	1.83	2.03	0.05
0.50	100	20K	160	9.16	3.91	0.12
1.01	128	25K	256	18.31	7.13	0.20
10.05	195	64K	512	182.81	24.38	0.76
101.47	315	160K	1024	1845.70	88.59	3.08

Table from ZeRO-Infinity

Idea

- Analysis of memory consumption
  - Model states: parameters, gradients, optimizer states
    - Problem: DP replicates model states across all devices
    - Solution: ZeRO-DP, remove memory redundancies in DP

Idea

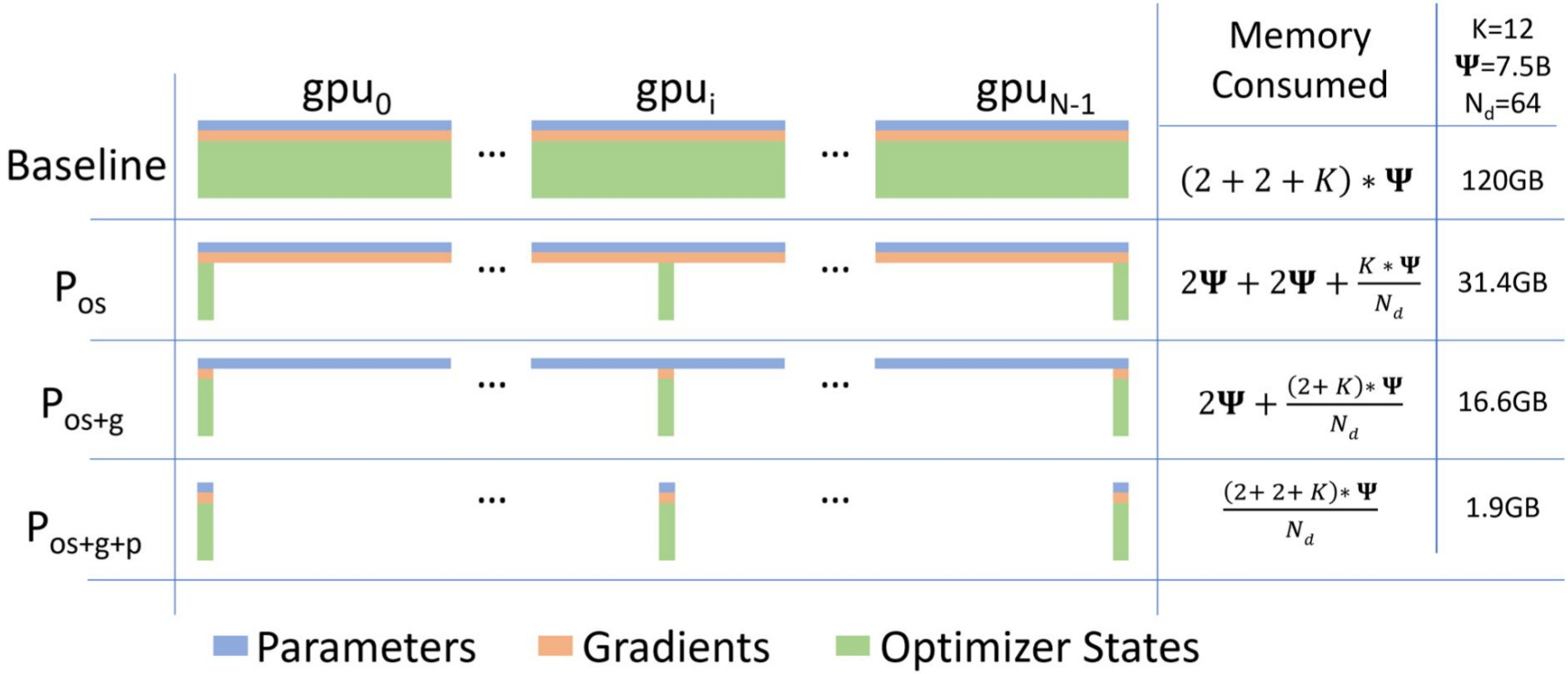
- Analysis of memory consumption
  - Model states: parameters, gradients, optimizer states
    - Problem: DP replicates model states across all devices
    - Solution: ZeRO-DP, remove memory redundancies in DP
  - Activation, temporary buffers, fragmented memory
    - Solution: ZeRO-R
      1. Avoid duplication in activation checkpoints of MP
      2. Set appropriate size for temporary buffers
      3. Prevent memory fragmentation

Method 1: ZeRO-DP

- Memory requirement from model states (Ψ := # parameters)
  - Parameters (fp16): 2Ψ
  - Gradients (fp16): 2Ψ
  - Optimizer states: e.g. Adam 12Ψ
    i. Parameters (fp32): 4Ψ
    ii. Momentum (fp32): 4Ψ
    iii. Variance (fp32): 4Ψ

- Approach: partition each of them to all DP processes

Method 1: ZeRO-DP



Method 1: ZeRO-DP

- Communication analysis (Ψ := # parameters)
  - Baseline DP: one all-reduce, 2Ψ
  - P_{os+g}: 2Ψ
    i. Scatter-reduce on gradients: Ψ
    ii. All-gather on updated parameters: Ψ

Method 1: ZeRO-DP

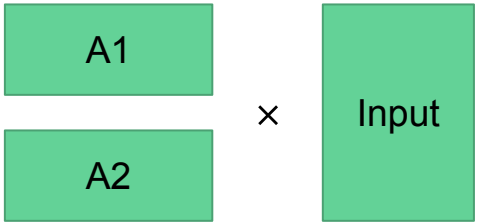
- Communication analysis (Ψ := # parameters)
  - Baseline DP: one all-reduce, 2Ψ
  - P_{os+g}: 2Ψ
    i. Scatter-reduce on gradients: Ψ
    ii. All-gather on updated parameters: Ψ
  - P_{os+g+p}: 3Ψ (1.5x communication)
    i. All-gather on parameters for forward: Ψ
    ii. All-gather on parameters for backward: Ψ
    iii. Scatter-reduce on gradients: Ψ

Method 2: ZeRO-R

- P_a: Partitioned Activation Checkpointing
- C_B: Constant Size Buffers
- M_D: Memory Defragmentation

Method 2-1: Partitioned Activation Checkpointing

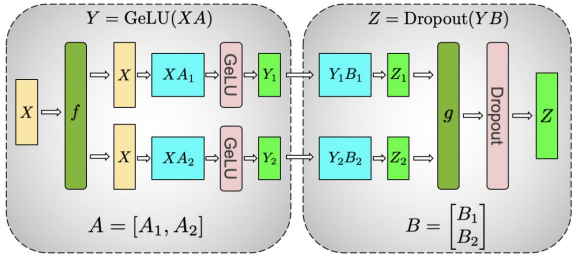
- MP usually requires replication of the activation
  - e.g. matrix multiplication



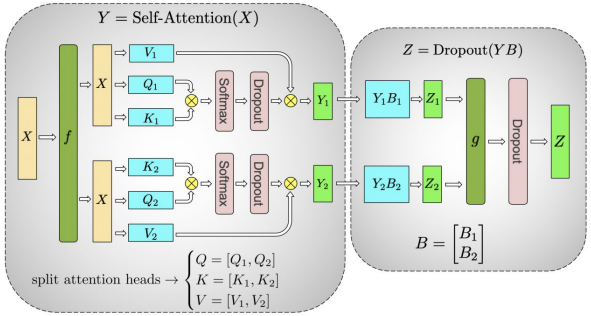
- Approach
  - Partition activation checkpoints across devices (P_a)
  - Offload activation to the CPU (P_{a+cpu}) if DP is the bottleneck
  - Can work together with rematerialization

Method 2-1: Partitioned Activation Checkpointing

- Communication analysis (*M* := message size of one input)
  - Baseline Megatron-LM: 12*M*
    i. Forward: 2 all-reduces 4*M*
    ii. Rematerialization: 2 all-reduces 4*M*
    iii. Backward: 2 all-reduces 4*M*



(a) MLP



(b) Self-Attention

Figure from Megatron-LM

Method 2-1: Partitioned Activation Checkpointing

- Communication analysis (*M* := message size of one input)
  - Baseline Megatron-LM: 12*M*
    i. Forward: 2 all-reduces 4*M*
    ii. Rematerialization: 2 all-reduces 4*M*
    iii. Backward: 2 all-reduces 4*M*
  - ZeRO-R P_a: 13*M* (<10% overhead)
    i. Baseline: 12*M*
    ii. All-gather on activation: *M*

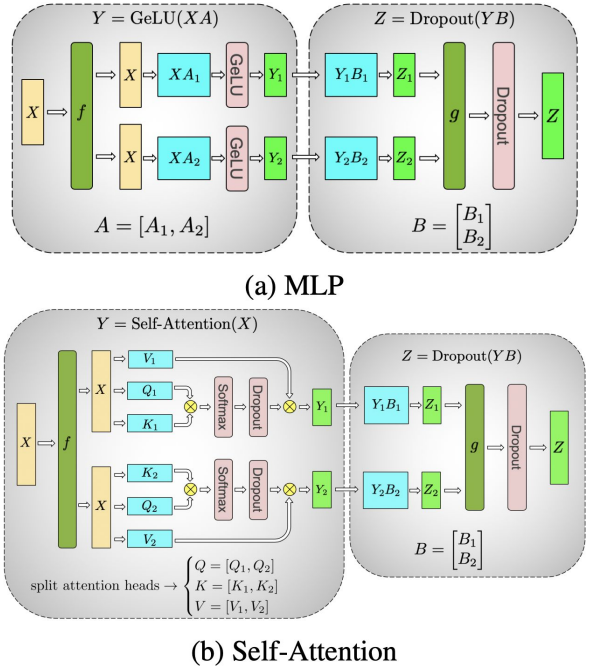


Figure from Megatron-LM

Method 2-2: Constant Size Buffers

- Example of temporary buffer: fuse all parameters into one buffer before all-reduce for higher bandwidth usage

- Approach (C_B)
  - Only allow constant-size buffer (not depend on model size)
  - Keep efficiency by setting the buffer size large enough

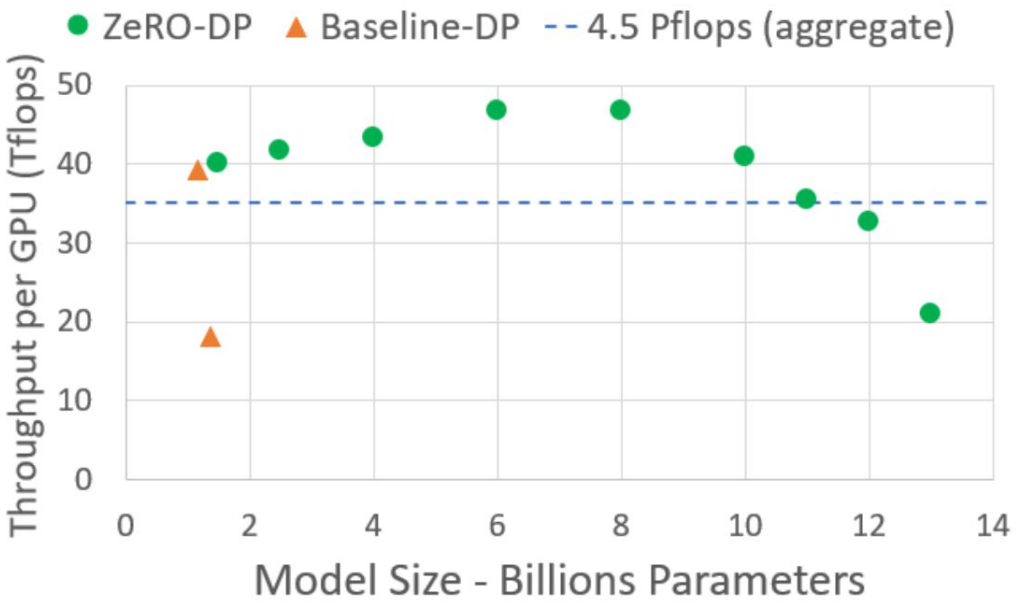
Method 2-3: Memory Defragmentation

- Memory life
  - Short lived memory: discarded activation
  - Long lived memory: checkpointed activation, model states
- Interleaving of short and long lived memory causes fragmentation

- Approach (MD): pre-allocate contiguous memory chunks for long lived memory

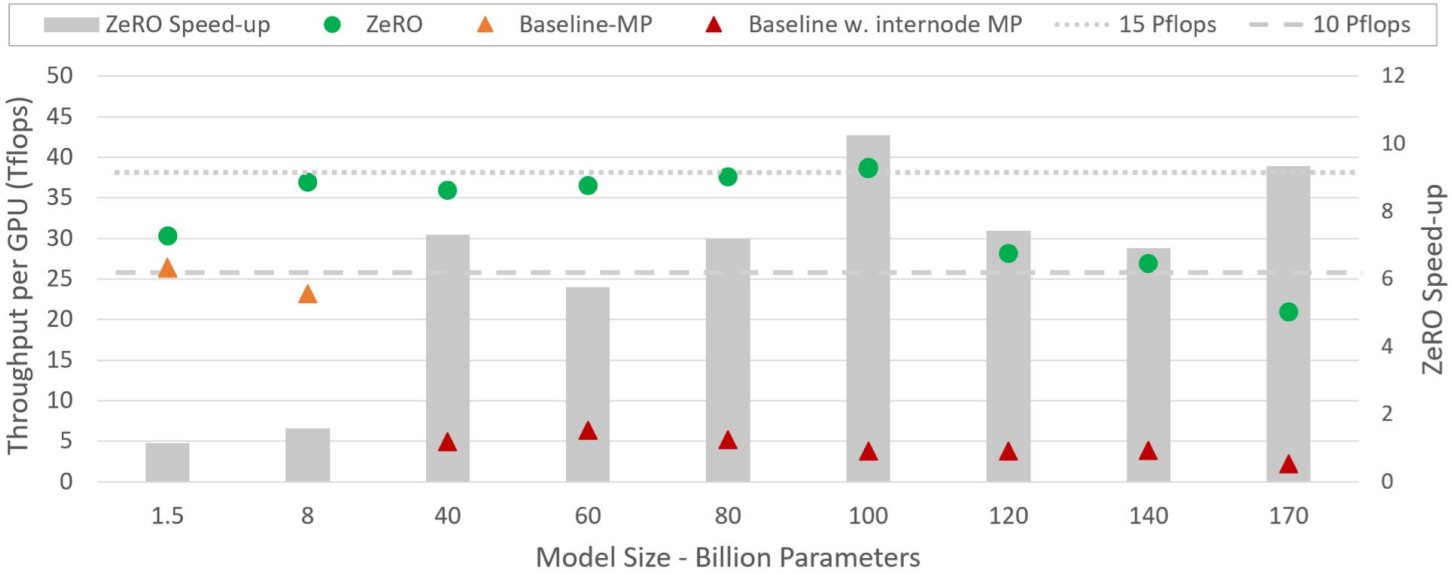
Experiments: ZeRO-DP vs. DP

- Support 10B-level model training without MP and PP
  (1.4B without ZeRO)
- Make large model training easier



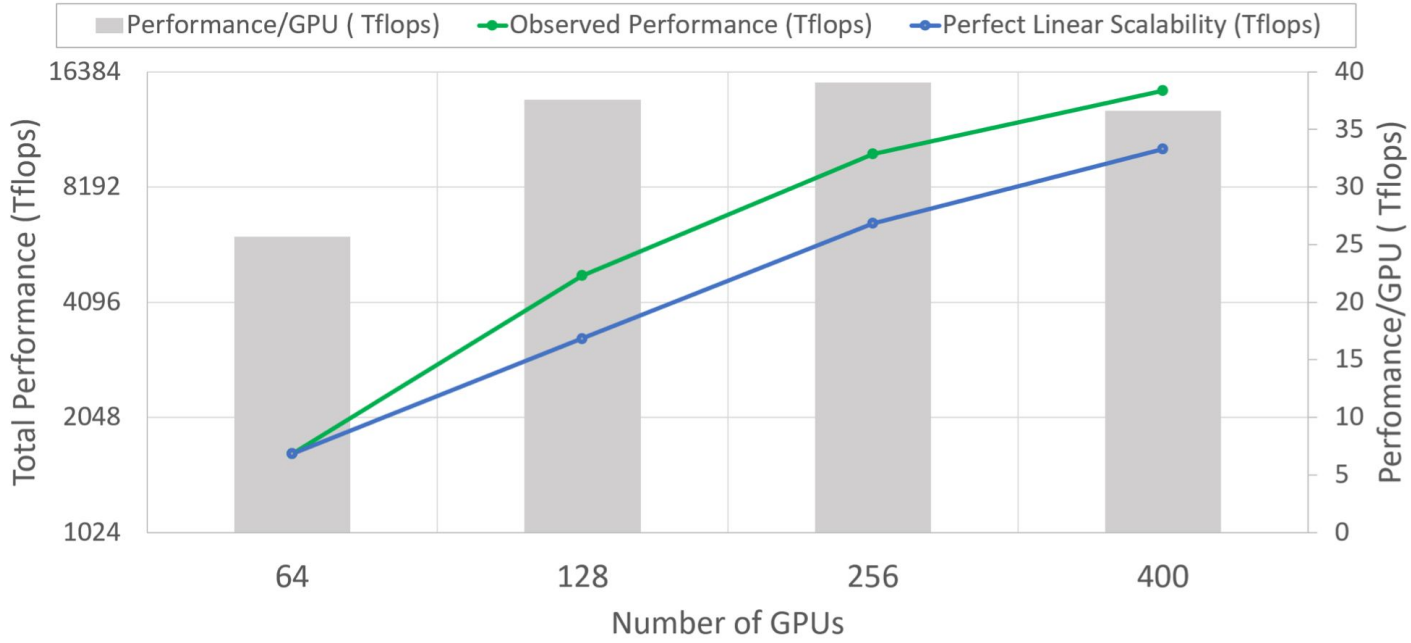
Experiments: ZeRO-DP & ZeRO-R vs. MP

- ZeRO avoids inefficient internode MP with ZeRO-DP
- Efficiently scale to 100B-level models



Experiments: ZeRO-DP & ZeRO-R

- Super-linear scaling in total performance



Experiments: Ablation Studies

	ZeRO-DP	*ZeRO*-R
1	P_{os}	C_B+M_D
2	P_{os}	C_B+M_D+P_a
3	P_{os+g}	C_B+M_D
4	P_{os+g}	C_B+M_D+P_a
5	P_{os+g}	C_B+M_D+P_{a+cpu}

- P_a effectively enlarge max model size
- For 100B-level models, the effect of offloading to CPU is noticeable
- Lower memory -> larger batch size -> better throughput per GPU

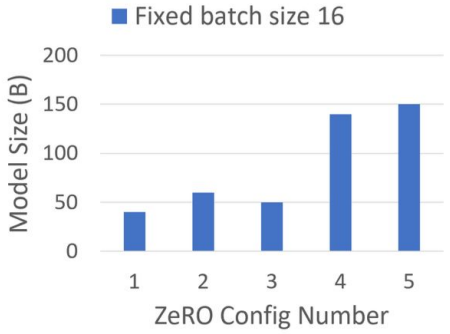


Figure 6: Max model size

Figure 7: Max cache allocated.

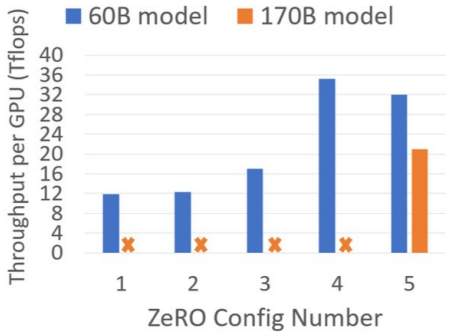


Figure 8: Throughput per GPU.

Application of ZeRO

- Turing-NLG (17B)

- Megatron-Turing NLG (530B)
  - 560 DGX A100 servers each with 8 NVIDIA A100 80G GPUs
  - 3D parallelism (8-way MP, 35-way PP, 16-way ZeRO-DP)
  - Combination of Megatron-LM with DeepSpeed (ZeRO)

Strengths of ZeRO

- ZeRO-DP completely resolves memory redundancies in DP, while largely preserving its communication efficiency
- ZeRO-R helps further reducing memory consumed by activation checkpoints
- ZeRO can be applied together with previous MP, PP and activation checkpointing techniques

Limitations of ZeRO

- To train a trillion-parameter model with ZeRO, the model states still has to fit in the total memory of all devices, i.e. the memory optimization is still upper bounded by the memory efficiency of MP
- ZeRO only considers reducing device memory consumption to make training giant models feasible, yet training speed is still limited by computing power (total FLOPS), e.g. training a trillion-parameter model can potentially take >100 days even with 1,000 state-of-the-art GPUs

Discussion

1. Is it possible to further reduce memory consumption in order to enable huge model training with fewer devices?

2. We have seen that search-based methods (TASO, GO, FlexFlow, etc.) prove to be beneficial for many ML system tasks. Is it feasible to apply some automatic search for improving ZeRO?

3. Previously model size tends to increase much faster than device performance. The current largest models already require top supercomputers for training. Given this situation, is it reasonable to further consider training models with trillions of parameters?

Thank you for listening!