ZeRO: Memory Optimization for Training Trillion-Parameter Models

- Samyam Rajbhandari, Jeff Rasley , Olatunji Ruwase, Yuxiong He

- Hemang Singh

The Problem: Why Do We Even Need ZeRO?

The Growing Model Size Problem

Model Evolution:

- BERT-Large (2018): 0.3 Billion parameters
- GPT-2 (2019): 1.5 Billion parameters
- Megatron-LM (2019): 8.3 Billion parameters
- T5 (2019): 11 Billion parameters
- Future Goal: 1 TRILLION parameters

The Challenge:

- These models don't fit in GPU memory!

Existing Solutions & Their Limitations

1. Data Parallelism (DP)

- Good: Fast, efficient computation
- Bad: Every GPU keeps complete copy → MEMORY WASTE
- Limit: Only 1.4B parameters on 32GB GPU

2. Model Parallelism (MP)

- Good: Splits model → saves memory
- Bad: Lots of communication between GPUs
- Bad: Slow across multiple nodes
- Limit: ~20B parameters efficiently

3. Pipeline Parallelism (PP)

- Complex to implement
- Needs huge batch sizes
- Affects training convergence

Where Does Memory Actually Go?

For 1.5B parameter GPT-2 model:

1. **Model States (Majority of Memory):**
    - Parameters (FP16): 3 GB
    - Gradients (FP16): 3 GB
    - Optimizer States (FP32): 18 GB
        - Parameters copy: 6 GB
        - Momentum: 6 GB
        - Variance: 6 GB
    - TOTAL: 24 GB

1. **Residual States:**
    - Activations: 60 GB (→ 8 GB with checkpointing)
    - Temporary Buffers: 6 GB
    - Fragmented Memory: Variable

Total: 90+ GB for a "3GB" model!

ZeRO's Core Insight

- Traditional Approach: Keep EVERYTHING on EVERY GPU
  - Wasteful: 64 GPUs = 64 complete copies

- ZeRO's Insight: Not everything is needed all the time!
- Example - Layer-by-Layer Processing:
  - Forward Pass Layer 3 → Only need Layer 3 parameters
  - Backward Pass Layer 5 → Only need Layer 5 parameters

Solution:

- Partition everything across GPUs
- Communicate dynamically when needed

ZeRO-DP Stage 1 - Optimizer State Partitioning

Stage 1: Pos (Optimizer State Partitioning)

Without ZeRO (4 GPUs):

GPU1	GPU2	GPU3	GPU4
Opt[ALL] 18GB	Opt[ALL] 18GB	Opt[ALL] 18GB	Opt[ALL] 18GB

Total per GPU: 18 GB × 4 = 72 GB wasted!

With ZeRO Stage 1 (4 GPUs):

GPU1	GPU2	GPU3	GPU4
Opt[1/4] 4.5GB	Opt[2/4] 4.5GB	Opt[3/4] 4.5GB	Opt[4/4] 4.5GB

Result: 4x Memory Reduction

Communication: Same as baseline

ZeRO-DP Stage 2 - Gradient Partitioning

Stage 2: Pos+g (Add Gradient Partitioning)

- Key Insight: Each GPU only needs gradients for ITS parameters!
- During Backward Pass:
  - Layer N completes → Gradients ready → Reduce-Scatter: Send each gradient partition to the GPU that owns it →Other GPUs discard this gradient
- Memory Saved:
  - Stage 1: 4Ψ (params+grads) + KΨ/Nd (opt states)
  - Stage 2: 2Ψ (params) + 14Ψ/Nd (grads+opt states)

- Result: 8x Memory Reduction (when Nd is large)
- Communication: Still same as baseline!

ZeRO-DP Stage 3 - Parameter Partitioning

Stage 3: Pos+g+p (Add Parameter Partitioning)

- **The Full Solution:**
  - Each GPU stores only 1/Nd of:
    - Optimizer States and Gradients and Parameters
- **During Forward/Backward:**
  - Broadcast needed parameters from owner
  - Use them for computation
  - Discard immediately after use
  - Move to next layer
- **Memory Reduction: Linear with GPU count!**
  - 64 GPUs: 64x memory reduction
  - 1024 GPUs: 1024x memory reduction
- **Trade-off:**
  - Communication: 1.5x baseline (Still worth it!)

ZeRO-R - Optimizing Residual Memory

Three Optimizations:

1. Pa: Partitioned Activation Checkpointing
    - Problem: Model Parallelism duplicates activations
    - Solution: Partition activations across MP GPUs
    - Benefit: 16x reduction (with MP degree 16)
    - Optional: Pa+cpu - Offload to CPU memory
    - When: Extremely large models

1. CB: Constant Size Buffers
    - Problem: Buffers grow with model size
    - Solution: Fixed-size efficient buffers
    - Benefit: Predictable memory usage

1. MD: Memory Defragmentation
    - Problem: Fragmented memory → OOM errors
    - Solution: Pre-allocated contiguous buffers
    - Benefit: 30%+ more usable memory

Experimental Results - Model Size

- Maximum Trainable Model Size:
  - Baseline (Megatron-LM):
    - Single Node (16 GPUs): 20B parameters
    - Multi-Node: Efficiency collapses
  - ZeRO-100B:
    - 400 GPUs: 170B parameters
    - Efficiency maintained!
- 8.5x Larger Models!

- Real Models Enabled:
  - 13B without Model Parallelism (vs 1.4B baseline)
  - 100B+ with Model Parallelism
  - Theoretical: 1T with 1024 GPUs

Experimental Results - Speed & Efficiency

- Throughput Comparison (100B model):
  - ZeRO: 38 TFlops/GPU
  - Baseline: <5 TFlops/GPU
- Aggregate Performance:
  - 15 Petaflops sustained (400 GPUs)
    - = 15 million billion calculations/second!
    - = 30%+ of theoretical peak
- Super-Linear Speedup (60B model):

64 GPUs	128 GPUs	1256 GPUs
Baseline	2.2x Faster	4.8x Faster

- Why Super-Linear?
- More GPUs → Less memory per GPU → Larger batches possible → Better GPU utilization

Strengths & Limitations

Strengths

- Massive memory savings
- Maintains DP simplicity
- Compatible with MP
- Enabled trillion-scale models

Limitations

- Runtime complexity
- Pp needs more bandwidth
- Compute cost still high
- Complex scheduling

Final Takeaway

- ZeRO eliminates all major sources of GPU memory redundancy
- Partitions optimizer states, gradients, parameters, and activations
- Achieves near-linear memory scaling with number of GPUs
- Keeps Data Parallel simplicity with far better efficiency
- Enables 100B–1T parameter training on commodity clusters
- Outperforms Model Parallelism without its communication pain
- Shifts the bottleneck from memory → compute
- Foundation behind modern large-scale systems (DeepSpeed, MT-NLG, etc.)

Open Questions & Future Directions

Open Questions & Future Directions

- How far can Stage 3 scale before communication dominates?
- Can ZeRO handle trillion-scale models efficiently across slower interconnects?
- What's the optimal balance of DP + MP + PP with ZeRO?
- How can ZeRO integrate with sequence parallelism and MoE architectures?
- Can optimizer/activation offload be made hardware-aware and adaptive?
- How does extreme batch scaling affect convergence and stability?
- Can ZeRO reduce FLOPs, not just memory?

Thank You !