SpareTrain: Fault-tolerant LLM Training via Low-Cost Dual Modular Redundancy

Rihae Park, Yeonjae Kim, Seung Yul Lee, Yeonhong Park, Jae W. Lee
Seoul National University

Paper

Silent Data Corruptions (SDC): A Real Threat in Large Model Training

What is Silent Data Corruption (SDC)?

- A hardware-induced computation error that escapes built-in detection.

SDC in Accelerators

- While memory is protected by the widespread use of ECCs, the logic units of accelerators remain unprotected and vulnerable. This vulnerability becomes more severe in large-scale clusters.

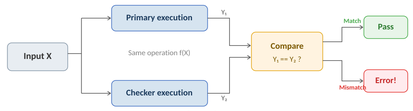
SDC in LLM training

- SDCs can silently corrupt model weights, causing drift from ground-truth—or even zero test accuracy on downstream tasks.
- Contrary to the assumption that transient errors are harmless in DNN training, they meaningfully degrade model quality.
- At the scale and cost of modern LLM training clusters, a corrupted run means massive financial loss.

Existing SDC Detection Approaches

Dual Modular Redundancy (DMR)

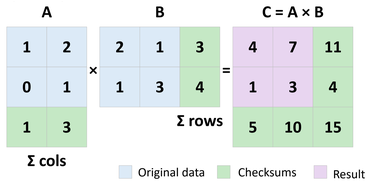
- Execute every operation twice and compare — the gold standard for SDC detection.



But 100% overhead

Algorithm-Based Fault Tolerance (ABFT)

- Checksum-based error detection — designed for high precision, but *impractical* with LLM training compute precision.



Approximate Approaches

- Protect only critical parts of the model or detect errors only when large anomalies appear.
- Use ML-based detectors, which require pretraining.

DMR remains the only reliable method, but it is too costly (2×).

How can we make DMR practical for large-scale LLM training?

Key Observations and Proposed Methods

1 Activation Checkpointing → Piggyback-DMR (P-DMR)

- Activation checkpointing: a memory-saving technique that drops forward activations and recomputes them in the backward pass.
- It is standard practice in large-scale LLM training: the freed memory enables larger batch sizes, which are critical for high training throughput.

Key observation: this recomputation is **inherently redundant** — the forward pass effectively runs twice.
Key idea: repurpose this inherent redundancy as a DMR mechanism itself.
→ **Piggyback-DMR (P-DMR)**

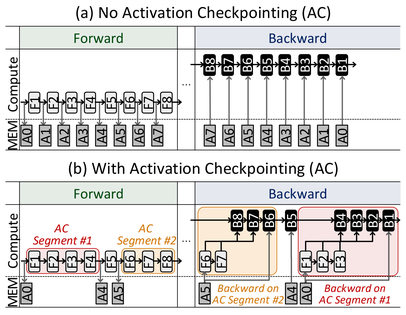


Figure 1. Activation Checkpointing — forward activations are dropped and recomputed during backward.

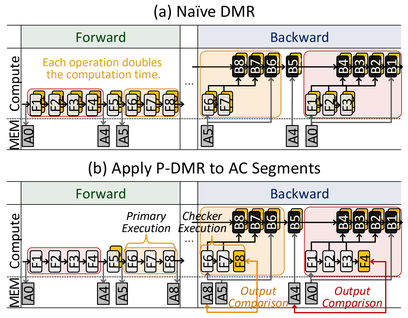


Figure 2. Naïve DMR vs. Piggyback-DMR — P-DMR reuses the AC recompute pass as the redundant execution.

2 GPU Idle Time → Deferred-DMR (D-DMR)

- LLMs are trained across many devices using various parallelism techniques, not on a single device.
- This introduces GPU compute idle time from inter-device communication and pipeline bubbles.

Key observation: even when GPUs wait on other devices or are busy with communication, **compute cores stay idle**.
Key idea: hide DMR execution inside these idle windows.
→ **Deferred-DMR (D-DMR)**

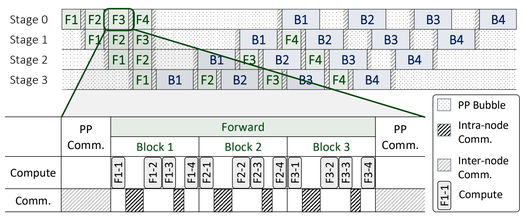


Figure 3. Compute & communication timeline — GPU compute cores stall during inter-device communication and pipeline bubbles.

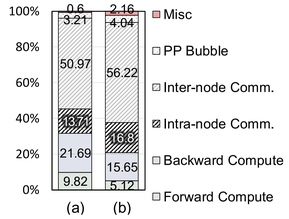


Figure 4. Training-time breakdown — idle time is a large fraction of total step time in modern LLM training. (a) Mistral-Large and (b) Llama-4-Scout.

How to utilize P-DMR and D-DMR to minimize DMR cost? — SpareTrain

SpareTrain

- Assigns every *op* to one of **four sets**: P-DMR, D-DMR$_{inter}$, D-DMR$_{intra}$, or Naïve-DMR.
- Two planners work together: a **Static Planner** (*offline*) and a **Dynamic Planner** (*online*).

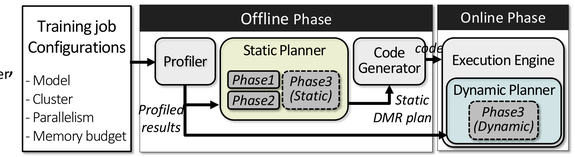


Figure 5. SpareTrain overview

- **Offline:** Profiler collects AC config, *op* time/memory, idle windows → Static Planner builds a DMR plan → Code Generator rewrites training code.
- **Online:** Execution Engine runs it; Dynamic Planner handles MoE operations at runtime.

3-Phase DMR Planning

1. Plan P-DMR

- Default: P-DMR for every AC-recomputed *op*.
- Exception: if 2 × Recompute > Additional Comm., fall back to naïve DMR.

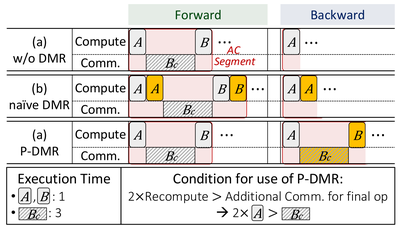


Figure 6. Exception case of AC segment where naïve DMR outperforms P-DMR.

2. *Coarse-grained* D-DMR

- Introduce Activation-Gradient Checkpointing (AGC) to replay grouped *ops* while keeping only boundary tensors in memory, and defer AGC segments into PP communication windows.

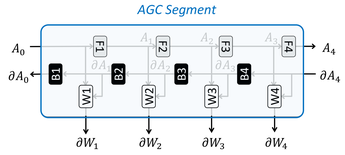


Figure 7. Concept of Activation Gradient Checkpointing (AGC)

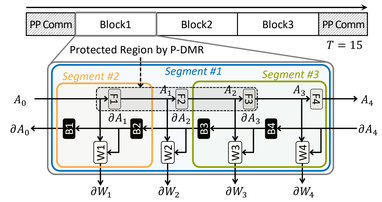


Figure 8. AGC segment selection process.

Constraints
- Memory Headroom (per Block) = 6
- Time Slack (per Block) = 5

	Segment Candidates	Step 2 Memory	Step 2 Execution Time	Step 2 Gain	Step 3	Step 4
Step 1	#1	8 > 6	12	9		
	#2	5	4	3		
	#3	6	6 > 5	5		

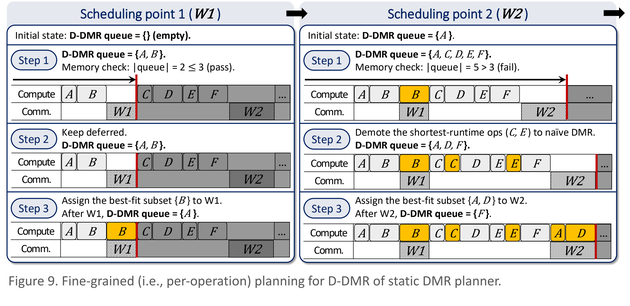


Figure 9. Fine-grained (i.e., per-operation) planning for D-DMR of static DMR planner.

3. *Fine-grained* D-DMR

- Manage remaining ops via a D-DMR queue, assigning them per-*op* into communication windows.
- At each scheduling point: 1) enqueue unassigned *ops*; 2) if over memory, demote shortest-runtime *ops* to Naïve-DMR; 3) assign a best-fit subset to the window.

MoE: Phase 3 runs at runtime via the dynamic planner.

Results — Training Throughput

- Evaluated on Llama-3-70B, Mistral-Large, Llama-4-Scout across 80 / 94 / 141 GB memory budgets on H200 clusters.
- 3–14% overhead over No-DMR (unprotected run).
- 12–35% higher throughput than Naïve-DMR.

Now, LLM training can achieve complete SDC detection at minimal cost!

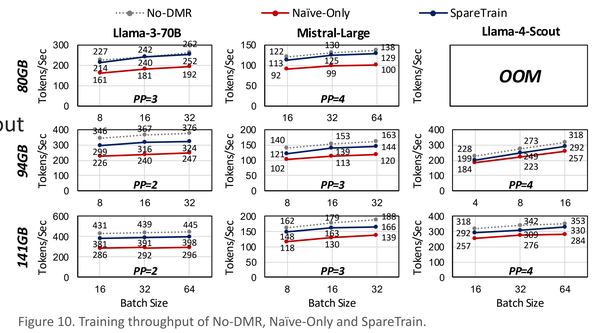


Figure 10. Training throughput of No-DMR, Naïve-Only and SpareTrain.