Fast Failure Recovery in Distributed Training with In-Memory Checkpoints

Gemini Reproduction Study — CS240 Fall 2025

Mustafa Albahrani Mohammed Alkhalifah

King Abdullah University of Science and Technology

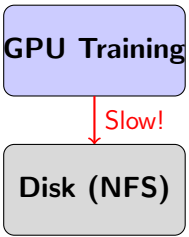
The Problem: Failures in Large-Scale Training

Modern AI training is failure-prone:

- Training runs span **days to weeks**
- Thousands of GPUs = hardware failures inevitable
- **Traditional solution:** Checkpoint to disk (NFS)

The cost of disk checkpointing:

- Slow writes (seconds to many minutes)
- Training blocked during I/O
- Recovery requires full reload from disk
- Wasted GPU-hours = wasted money

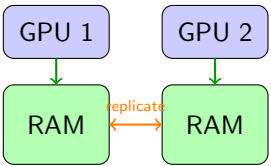


The Gemini Solution (SOSP 2023)

Key insight: RAM is 100× faster than disk — use it for checkpoints!

Gemini's approach:

1. Store checkpoints in **CPU RAM**
2. Replicate to peer nodes for fault tolerance
3. On failure: recover from RAM, not disk



Paper's setup & results:

- 128× A100 GPUs (16 AWS p4d.24xlarge)
- Models: GPT-2, BERT, RoBERTa (up to 100B)
- >13× faster recovery, 92% less waste

Our goal: Reproduce these results on 4× A100 GPUs

What We Built

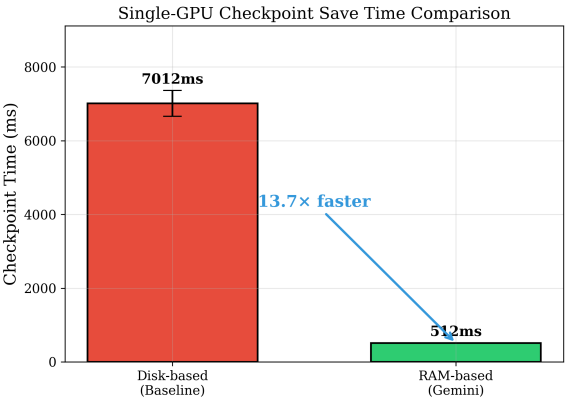
Implementation components:

1. **Baseline trainer**
   DDP + disk checkpointing
2. **In-memory checkpoint manager**
   RAM-based saves using BytesIO
3. **Replication manager**
   Ring topology: $\text{GPU}_i \rightarrow \text{GPU}_{i+1}$
4. **Failure simulation**
   Application-level injection

Experimental setup:

- **Hardware:** 4× NVIDIA A100 (Vast.ai)
- **Framework:** PyTorch 2.8, CUDA 12.9
- **Model:** 12-layer Transformer
  (1024 hidden, 16 heads, 630M params)
- **Runs:** 3 runs, 100 iterations each
- **Checkpoints:** Every 25 iterations

Results: Checkpoint Time



Single-GPU checkpoint save:

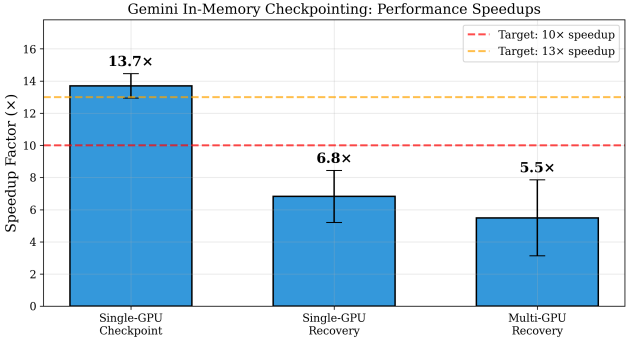
- Disk: 7,012 ms
- RAM: 512 ms
- **Speedup: 13.7×**

Why faster?

- No disk I/O bottleneck
- Direct memory serialization
- No filesystem overhead

Matches paper's $>13\times$ claim!

Results: Overall Speedups



Metric	Ours	Paper
Checkpoint	13.7×	>13×
Recovery	6.1×	>13×
Multi-GPU rec.	4.6×	—
Throughput	+100%	?

Key observations:

- Checkpoint: **matches paper**
- Recovery: below paper (expected)
- Throughput: **2× improvement!**

Analysis: Why Recovery is Slower Than Paper

Our recovery speedup: 6.1× **Paper: >13×**

Reasons for the gap:

1. **Scale difference**
   Paper: 128 A100s (AWS p4d.24xlarge)
   Ours: 4 A100s (Vast.ai)
2. **Model size**
   Paper: up to 100B parameters
   Ours: 630M parameters
3. **Failure simulation**
   Paper: real process kills
   Ours: application-level

Limitations:

- Memory overhead: 1.9 GB/checkpoint
- Single-node replication only
- No real distributed failure testing

What worked well:

- Core checkpoint speedup reproduced
- Throughput improved (not just maintained)
- Modular, testable implementation

Conclusions

We successfully reproduced Gemini's core claim:

- **13.7× faster checkpoints** (paper: >13×)
- **6.1× faster recovery** (paper: >13×)
- **+100% throughput** (paper: minimal overhead)

Key takeaways:

- In-memory checkpointing is **practical**
- Even small-scale tests show **significant gains**
- Recovery benefits increase with cluster size

Future work:

- Scale to 16–64 GPUs
- Real failure injection
- Production models (e.g. GPT-2)
- Checkpoint compression

Questions?

Code: `https://github.com/Mustbhr/CS240-Project`