CS240 – Project Proposal

Fall 2025

Project Title: Fast Failure Recovery in Distributed Training with In-Memory Checkpoints (Gemini Reproduction)
Student Name(s): Mustafa Albahrani, Mohammed Alkhalifa
Email: mustafa.albahrani@kaust.edu.sa, mohammed.alkhalifa@kaust.edu.sa
Type of Project: Reproduction study

2. Motivation and Project Goal

Modern large-scale AI training runs across thousands of GPUs for days or weeks, making hardware and network failures inevitable. Traditional checkpointing to slow persistent storage creates large recovery delays and wasted GPU-hours. The **Gemini system (SOSP 2023)** proposes using aggregated CPU memory as a fast, distributed checkpoint tier, achieving near-instant recovery while maintaining throughput.

This project aims to reproduce Gemini's key result: *¿13× faster failure recovery with minimal overhead*, and to demonstrate the feasibility of in-memory checkpointing in a scaled-down, academic cluster environment.
Targeted Traits: Reliability   Scalability

3. Project Details

Goal: Reproduce the core Gemini system that achieves fast, fault-tolerant recovery in distributed deep learning by storing and replicating model checkpoints in RAM across training nodes.
Architecture:

- **Worker Agents** on each node handle local checkpoint capture and replication.
- A simplified **Root Agent** coordinates recovery and failure detection.
- Checkpoint shards are replicated between nodes using a group-placement strategy ($m = 2$).
- Failures are injected to measure recovery latency vs. a baseline NFS-based checkpointing setup.

Technologies: Python, PyTorch, DeepSpeed (ZeRO-3), NCCL, etcd, CUDA.
Dataset: Scaled-down Wikipedia-en corpus from Hugging Face (`datasets.load_dataset("wikipedia", "20220301.en")`).
Hardware: 4 multi-GPU nodes (e.g., IBEX or AWS).

4. Plan and Milestones (around 6 weeks)

Week	Milestone	Deliverable
1	Environment setup (PyTorch + DeepSpeed on 4 nodes); run baseline training	Working distributed training job; baseline metrics (throughput, NFS recovery)
2–3	Implement in-memory checkpointing (local + replicated); add recovery logic	Checkpoint creation/loading from RAM; replication between nodes
4–5	Failure injection + automated recovery; evaluate recovery latency	Measured “wasted time” comparison vs. NFS baseline; plots and logs
6	Final integration + report	Demonstration, slides, and final report

5. Expected Outcomes

We will measure:

- **Recovery latency** (time from failure to resumed training)
- **Training throughput** with vs. without checkpointing
- **Wasted time reduction** compared to a persistent storage baseline

Success is shown if the in-memory version achieves a 10–13× faster recovery with minimal throughput loss.

6. Resources Needed

- KAUST IBEX cluster ($\geq 4 multi - GPU nodes$)

- CUDA PyTorch, DeepSpeed
- etcd for coordination
- Wikipedia-en dataset (via Hugging Face)

7. Paper Details (Reproduction Study)

Paper Title: *Gemini: Fast Failure Recovery in Distributed Training with In-Memory Checkpoints*
Venue: SOSP 2023 **Year:** 2023
Link: https://dl.acm.org/doi/10.1145/3600006.3613145
Code: GitHub Repository (Artifact Evaluation)
Reproduction Goal: Replicate Gemini’s key experiment comparing **recovery latency and throughput** against persistent storage checkpointing, validating whether in-memory checkpointing yields fast recovery without throughput degradation:contentReference[oaicite:1]index=1.

8. Risk Assessment and Feasibility

Risks:

- Artifact code is tightly coupled to AWS environment $\rightarrow$ mitigate by time-boxing adaptation $(\leq 1.5weeks)andreimplementingMVRfromscratchifneeded.$
- Limited GPU access $\rightarrow$ coordinate early with IBEX admins or use small AWS budget.
- DeepSpeed integration complexity $\rightarrow$ limit to Python hooks and callbacks instead of engine modification.

9. References

[1] Wang et al., *Gemini: Fast Failure Recovery in Distributed Training with In-Memory Checkpoints*, SOSP 2023.

Gemini Artifact Repository, GitHub, 2023.