

1 INTRODUCTION

PROBLEM

Preempted training jobs restart on fresh nodes without local compiler caches, repeating compilation. In our deployment, a live shared cache caused competing writes, incomplete source reads, and publication failures.

WHAT WENT WRONG · SHARED LIVE CACHE



One shared live cache — competing writes

Triton: FileNotFoundError during os.replace amid competing writers.

CONTRIBUTION

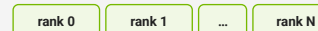
Workers compile locally. One designated rank atomically publishes a completed snapshot for restart, separating local compilation from shared publication.

2 METHODS AND RESULTS

Multi-node · FSDP2 hybrid sharding · regional torch.compile per layer

PROTOCOL / One publisher, many readers

1 TRAIN ON CURRENT NODES



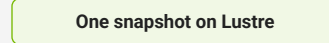
Node-local compiler caches

Each GPU holds one parameter shard but executes the layers on its batch. Missing graphs compile locally.



2 SAVE TO SHARED STORAGE

After step 1, rank 0 exports artifacts from paths it exercised.



A temporary file is atomically renamed to publish a completed snapshot.



3 RESTART ON NEW NODES



Import before compiled forward

Every rank checks the schema version, then reads the shared snapshot. Incompatible caches are rejected.

Snapshot scope: paths rank 0 exercised in step 1. Unseen paths or shapes may still need compilation.

1.26×

Transformer-trunk
microbenchmark speedup

7.6%

Lower warm median step
time 0.3947 → 0.3647 s

20.4 MB

snapshot imported on
16/16 ranks in 0.15 s
after request

3 CONCLUSION

Per-rank median	Cold	Restart	Saved
Cache hits (AOT / FX)	0% / 0%	100% / 100%	all eligible
Compile time, step 1	18.8 s	8.7 s	10.1 s (54%)
Time to first step	31.5 s	19.7 s	11.8 s (37%)
Snapshot import	—	20.4 MB, 0.15 s	16 / 16 ranks

Model: A multimodal model with a language backbone (0.5B) and an image encoder (0.3B) · FSDP2 hybrid sharding, 16 ranks · torch.compile per layer · torch 2.8
AOT = AOTAutograd cache · FX = TorchInductor FXGraphCache

In a 16-rank cold/restart A/B, all eligible graphs hit on every rank. Step-1 compilation fell 54%; time to first step fell 37%, with identical step-1 loss. Rank 0 behaved like ranks 1–15.

Bad snapshots fall back to cold compilation. Coverage—not hit rate—limits reuse.

Compiler artifacts are reusable optimizations, not correctness dependencies.

4 FUTURE WORK

1 Reuse artifacts from a small warm-up job

Generate artifacts in a single-node warm-up job for larger runs. Compile missing paths locally; measure coverage, startup latency, and total GPU-hours across scales.

2 Share artifacts across compatible experiments

Explore a trusted, read-only storage location for compatible model, compiler, and hardware configurations. Keep writes local; evaluate reuse, safe refresh, and retention.

3 Reduce remaining startup overhead

Evaluate newer PyTorch caches for checkpointed graphs and tracing. Warm up representative forward/backward paths while awaiting data. Measure time to first real step, numerical consistency, and unintended recompilation.