LSAI Project

Team Members:
- George Manos:
- Luca Renna: luca.renna@infk.ethz.ch

Project Goal:

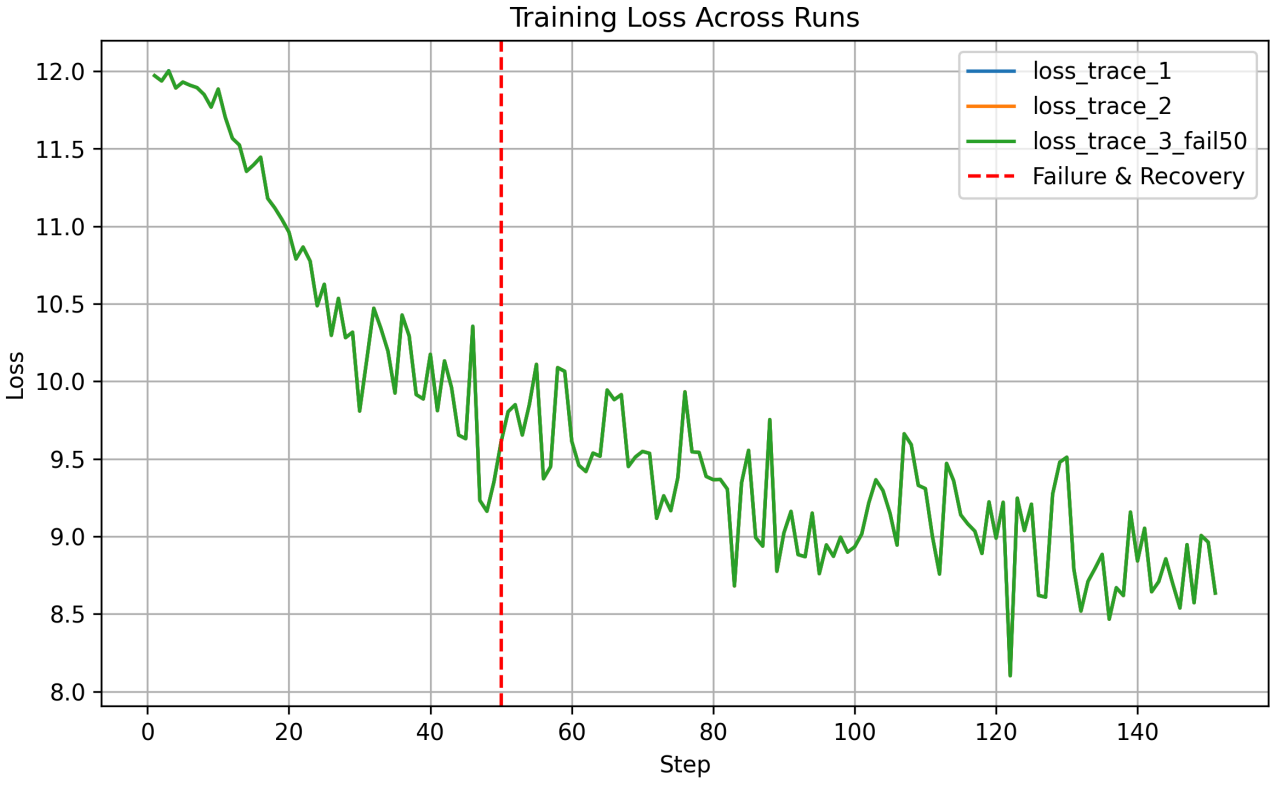
In this project we implement the intra-node version of the checkpointing mechanism outlined in "PCcheck: Persistent Concurrent Checkpointing for ML" (2025, Strati et al) and compare it against a naive baseline checkpointing implementation.

Baseline:

We implement the baseline using a standard pytorch.save, where we checkpoint the current training step, the model state, optimiser state, scheduler state and the dataloader state. The complete code implementation can be found under `/training/train_checkp.py`.

To test correctness, we launch two runs using seed=4, with 150 training steps and checkpoints at 50-steps intervals. We then launch a third runs with the same parameters but with a failure at training step 51. We then recover from the checkpoint and we resume training.

We plot and compare the losses in the following figure:



From the figure and the log files, we verify that the losses are exactly the same for a given seed and regardless of failure recovery.

Baseline Performance
For our checkpointing, we simulate training on a 32-layer Transformer with a 4096-dimensional model width, 32 attention heads sharing 8 key/value heads (grouped-query attention), a feed-forward dimension of 6144, Rotary Position Embeddings with $\theta = 500000$ and a maximum sequence length of 4096 tokens. The model parameters are updated with the AdamW optimiser.

We report the following times using Python's `perf_counter` timing. Note that we do not consider the time taken to transfer the model from host memory to device memory.

Checkpoint Saving Time: $\approx$ 35 seconds
Checkpoint Loading Time: $\approx$ 12 seconds

PCcheck Implementation

You can write here what you have done to adapt PCcheck to the cluster and other stuff that you deem relevant.