Rehabilitating over Recomputing: A Novel Failure Recovery Method for Large Model Training

Presenter: Hongliang Li

Zichen Wang[1], Hongliang Li[*1,2], Jie Wu[*3,4], Zhewen Xu[1], Hairui Zhao[1], Qi Tian[1], Haixiao Xu[5]

1 College of Computer Science and Technology, Jilin University, Qianjin street 2699, Changchun, 130012, China

2 Key Laboratory of Symbolic Computation and Knowledge Engineering of the Ministry of Education, Changchun, China

3 Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA

4 China Telecom Cloud Computing Research Institute, China

5 High Performance Computing Center, Jilin University, China

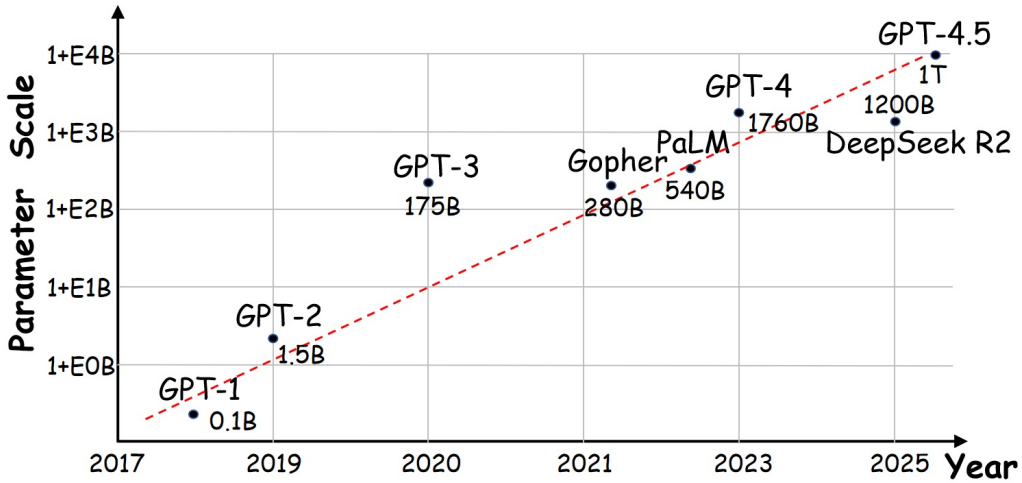
*Corresponding author

Outline

- Background and Motivation
- CPSR Framework and Checkpoint Interval Problem
- Evaluations

Background (1)

❖ Fault tolerance in large AI model training



Scaling law: bigger model, more data, better result

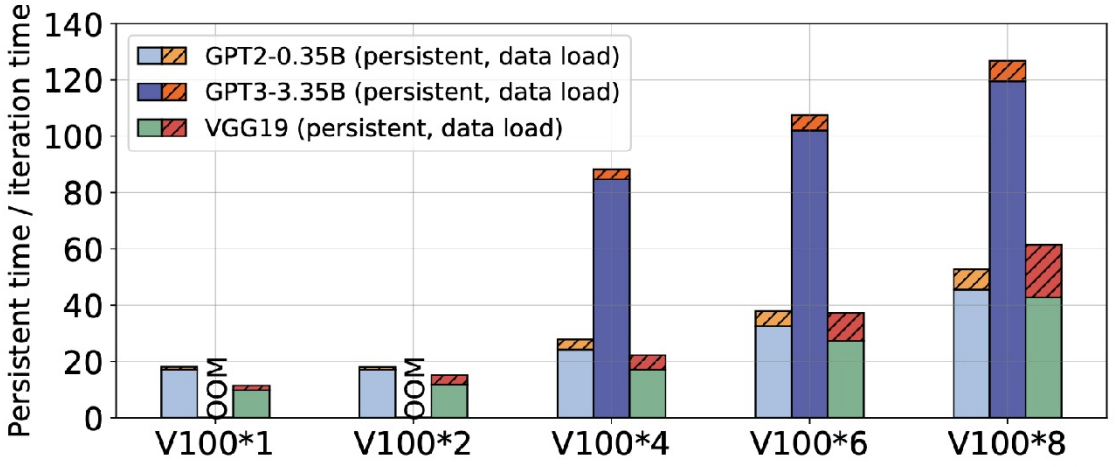
Long training time and larger parallel scale make the training process more vulnerable to perturbations.

- Training interruptions are frequent (LLaMA3 reported 466 interruptions during a 54-day training period)
- Checkpoint/Recompute mechanism, takes periodic "snapshots" of model states (parameter/optimizer) and metadata.
- Fault-tolerant cost can be noticeable (OPT-175B with CKPT interval of 3 hours, 178k GPU hours wasted, 13% of total)

Background (2)

❖ Checkpoint frequency is limited, causing high recovery cost

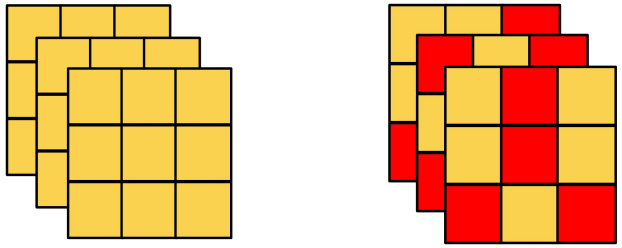
- Latest checkpointing methods reduced training interruptions but frequent checkpointing remains infeasible in practice
  - Resource competition (such as I/O bandwidth) affects checkpointing performance (both snapshot and persistent stages)
- Long interval (hours) leads to high recomputing cost in the event of a failure
- How to reduce recovery cost?

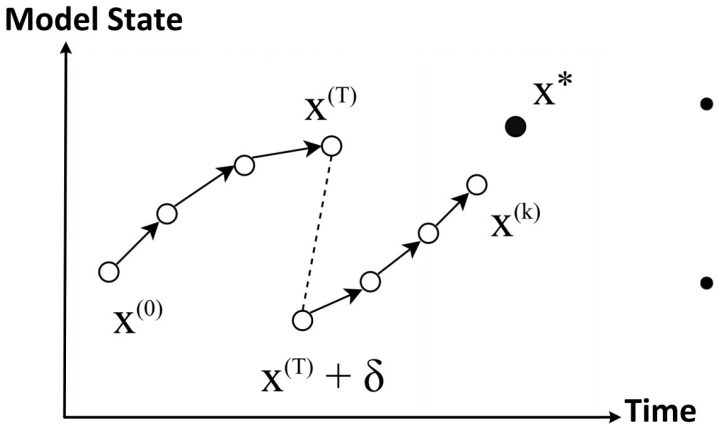


Motivation (1)

❖ Consistent or inconsistent fault tolerance

Consistent vs. inconsistent

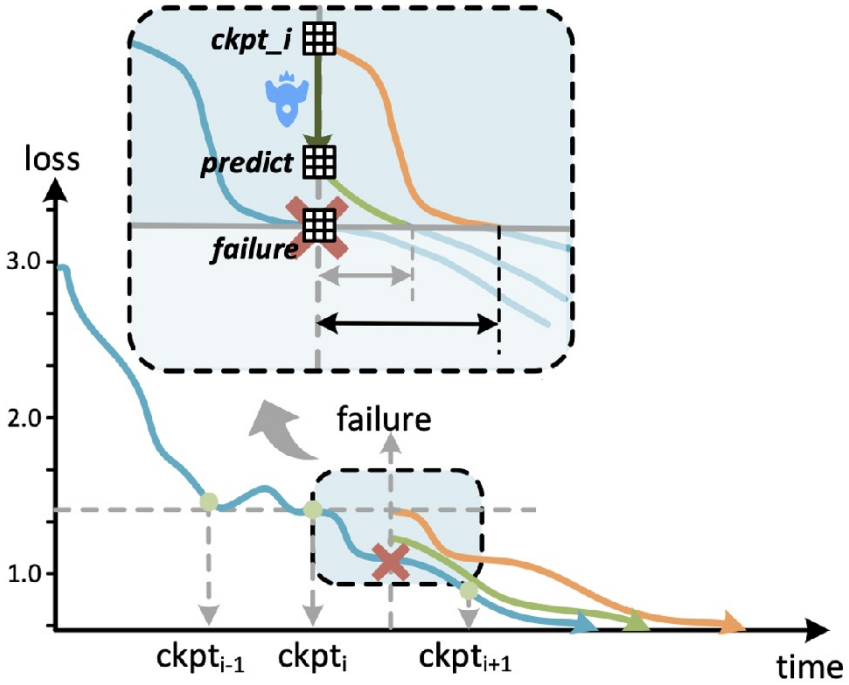


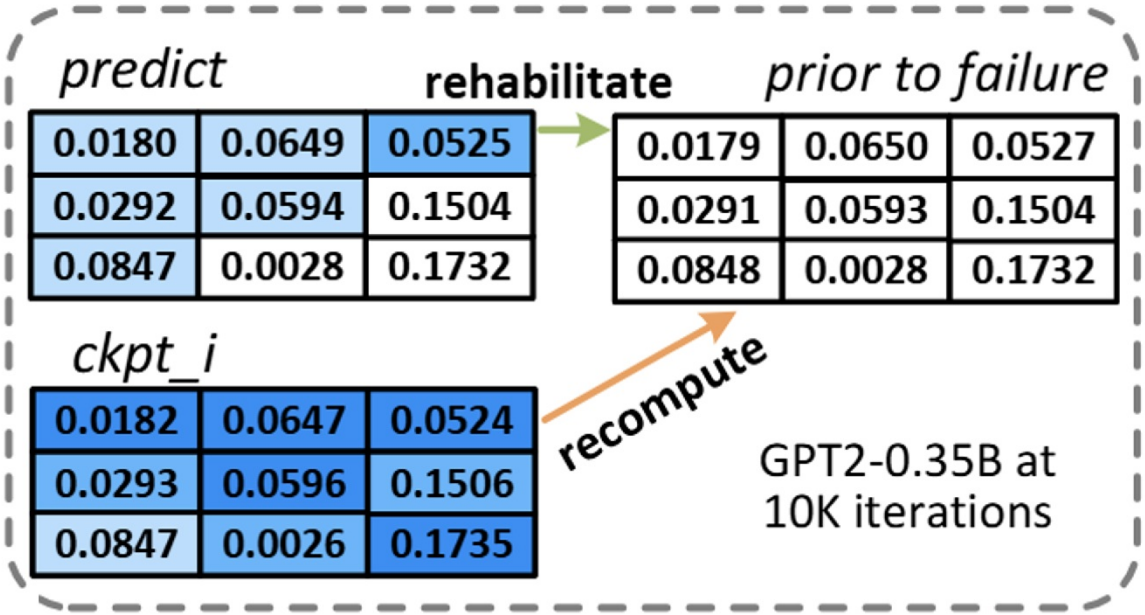


- Checkpoint/recompute: the recovered training state is kept consistent with the original state prior to a failure.
- DNN training is an iterative convergence process that can endure perturbations, such as model quantization, lossy recovery, stale model update, etc.
- The training process has inherent self-recovery capability.
- Minor inconsistencies of parameter states among parallel training tasks do not affect the global convergence properties.

Motivation (2)

- Is it possible to control and speed up the self-recovery?
  - Recomputing: from the latest checkpoint, redo the training
  - Rehabilitating: from the latest checkpoint, predict model states, self-recovery





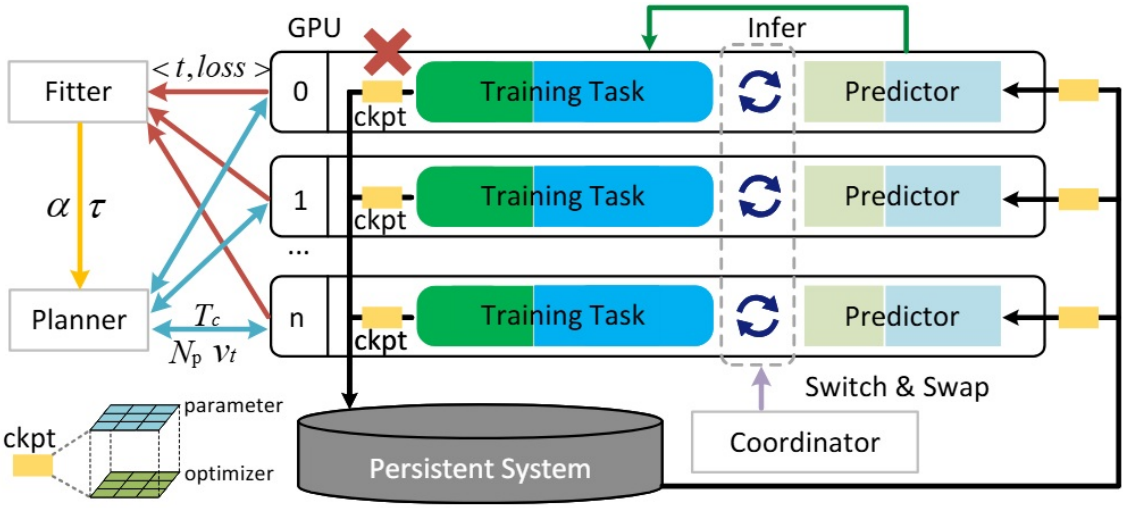
Challenges

i. Accurate model state prediction

ii. Better do this without adding too much pressure to the training job itself

iii. Decide checkpoint interval

Controlled Predicting-assisted Self-Recovery Overview



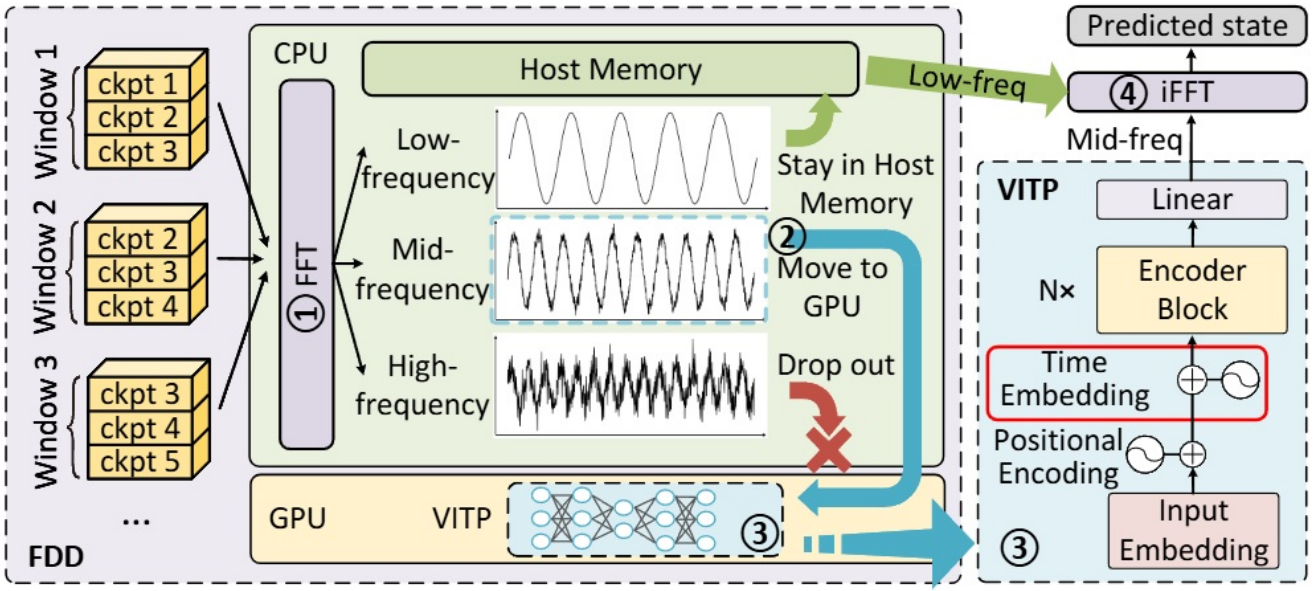
a) ***Task-level Predictors*** are trained in runtime, with local task checkpoints and can infer local training state (parameter & optimizer) from the last checkpoint to boost recovery.

b) ***Coordinator*** manages local device and execution flows between the training task and predictor tasks.

c) ***Fitter*** collects validation loss over time and characterizes loss curve coefficients, indicating convergence trajectory, which is used by ***Planner*** to compute optimal checkpoint plans.

Predictor: accurate model state prediction

❖ FFT-based Data Denoising (FDD) and Variable-Interval Temporal Predictor (VITP)



Data preparation on CPU side

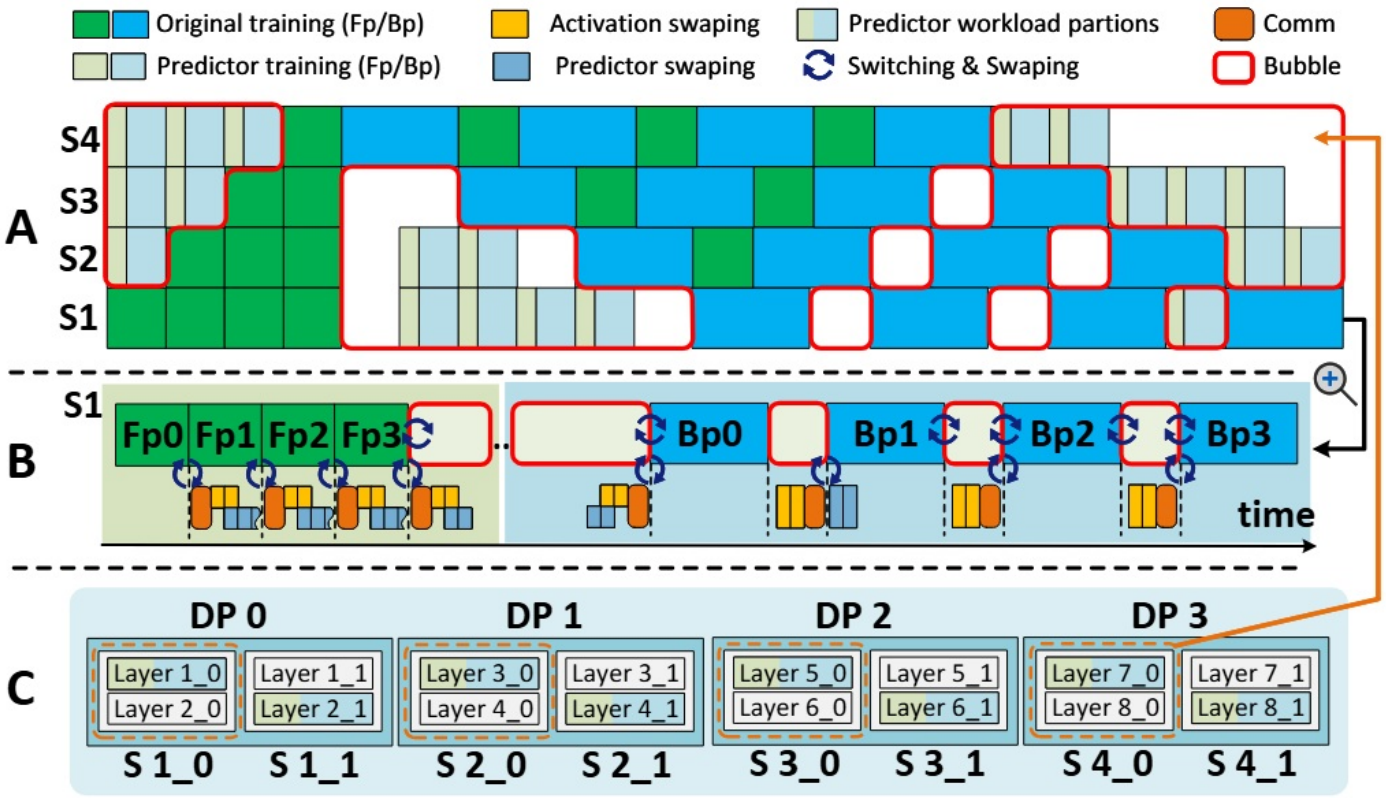
FDD: apply FFT to checkpoint data to separate frequency components:
high-frequency noise ❌
low-frequency baselines ✅
mid-frequency stable patterns ✅

VITP: uses dynamic time embeddings to handle variable intervals caused by non-uniform checkpoint intervals.

Organizing training data for Predictor:
Checkpoints are split along tensor dimension, arranged chronologically into a 2D array (tensor states and temporal), and partitioned into windows.

Coordinator: lightweight predictor training

❖ Near-zero overhead: Pipeline scheduling and utilizing "bubbles"



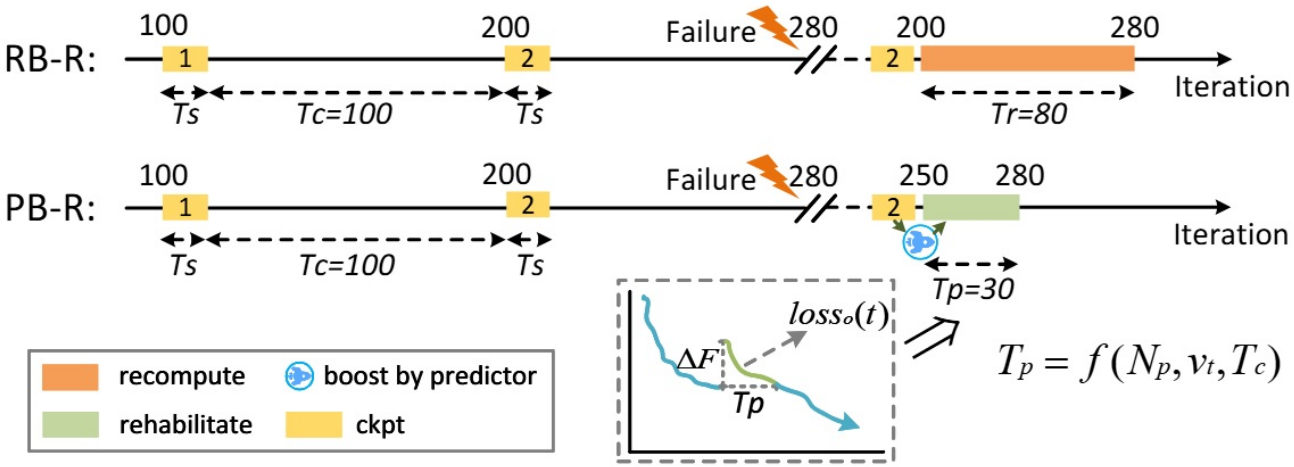
A: Predictor training task is scheduled to utilize pipeline bubbles.

B: Predictor is designed to be lightweight (<200MB) to enable frequent task switching.

C: Data parallelization further divides the Predictor training workload.

Planner: optimal checkpointing plan

- Quantitative modeling of Predictor accuracy, its trade-off relationship with checkpoint interval.



Tc： checkpoint interval

Ts： checkpoint cost

Tr： recomputing-based recovery cost

Tp： predicting-Based recovery cost

Predicting-based recovery speed correlates with Predictor accuracy, which depends on the number of parameters to predict (Np), the prediction sample size (determined by checkpoint interval Tc), and the training stage of the predicted parameters (vt).

PCIP: Predicting-Based Checkpoint Interval Problem

Given failure rate λ, checkpoint cost T_s, the amount of parameter affected by a failure N_p, and the second-order moment of the gradient v of a model, find a checkpoint interval T_c with minimum fault tolerance cost T_l:

$$minimum \qquad T_l = \sum_{i=0}^{\infty}(iT_s + T_p)P(t)d_t \qquad (1)$$

$$s.t. \qquad T_p = f(N_p, v_t, T_c) \qquad (2)$$

$$P(t) = \lambda e^{-\lambda t}, \lambda = 1/T_f \qquad (3)$$

$$F(y) \leq F(x) + \nabla(x)^T(y-x) + L\|y-x\|^2/2 \qquad (4)$$

$$F(y) \geq F(x) + \nabla(x)^T(y-x) + \mu\|y-x\|^2/2 \qquad (5)$$

Approach (1)

- Prediction Deviation **ΔF**: the variation of the loss value, between the predicted and the true training state.

$$\Delta F = \xi N_p e^{-\tau_1 \frac{t}{T_c} + \tau_2} / 2. \qquad (10)$$

  - τ_1, τ_2 are fitted coefficients of the Predictor loss curve

$$loss_p(t) = e^{-\tau_1 \frac{t}{T_c} + \tau_2}, \qquad (8)$$

  - Fitter: At the end of each training phase, the fitter is launched to approximate the loss descent trajectory (coefficients)

[1] Z. Bai, et al.,"Pipeswitch: Fast pipelined context switching for deep learning applications," in USENIX OSDI, 2020

Approach (2)

- Mapping prediction deviation to time-domain predicting-based recovery cost.

$$loss_o(k^{'}) = \underbrace{[(e^{-\alpha_1})^t \cdot e^{\alpha_2} + \Delta F]}_{predicting-assisted}(e^{-\alpha_1})^{k^{'}-t}. \qquad (12)$$

  - minor perturbation will not affect the original training trajectory
  - predicting-based recovery cost:

$$T_p = e^{\alpha_1 t_e - \alpha_2} \cdot (\xi N_p e^{-\tau_1 \frac{t}{T_c} + \tau_2})/2\alpha_1. \qquad (14)$$

Approach (3)

❖ Optimal Checkpoint Scheme

$$T_l = \sum_{i=0}^{\infty} \int_{i(T_c+T_s)}^{(i+1)(T_c+T_s)} (iT_s + \frac{e^{\alpha_1 t - \alpha_2 \cdot \Delta F}}{\alpha_1}) \lambda e^{-\lambda t} d_t. \quad (15)$$

- minimizes the fault-tolerant cost T_l:

$$T_c = \frac{\tau_1}{B + \lambda\sqrt{A/T_s}}, \quad (16)$$

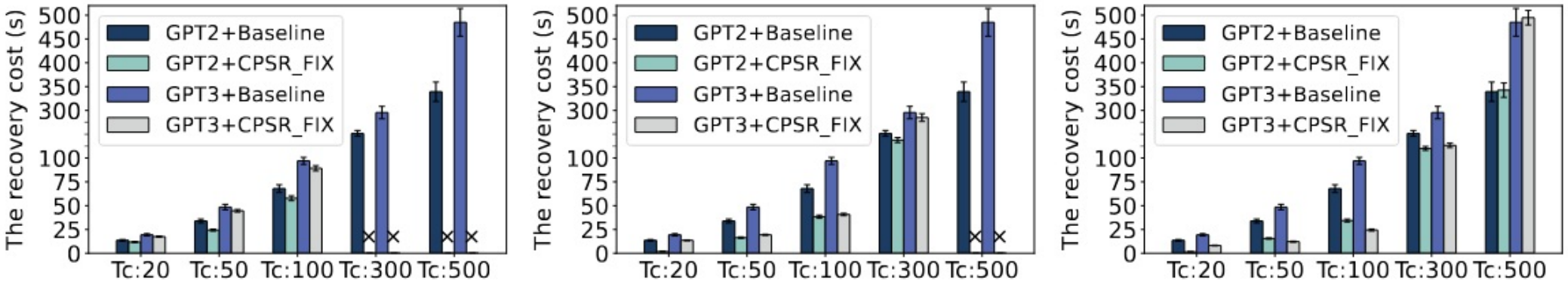
[1] Z. Bai, et al.,"Pipeswitch: Fast pipelined context switching for deep learning applications," in USENIX OSDI, 2020

Evaluations (recovery cost)

Testbed: NVIDIA A30 (24GB) × 8, PCIe4.0 (12GB/s).

Baselines: CheckFreq, PCcheck

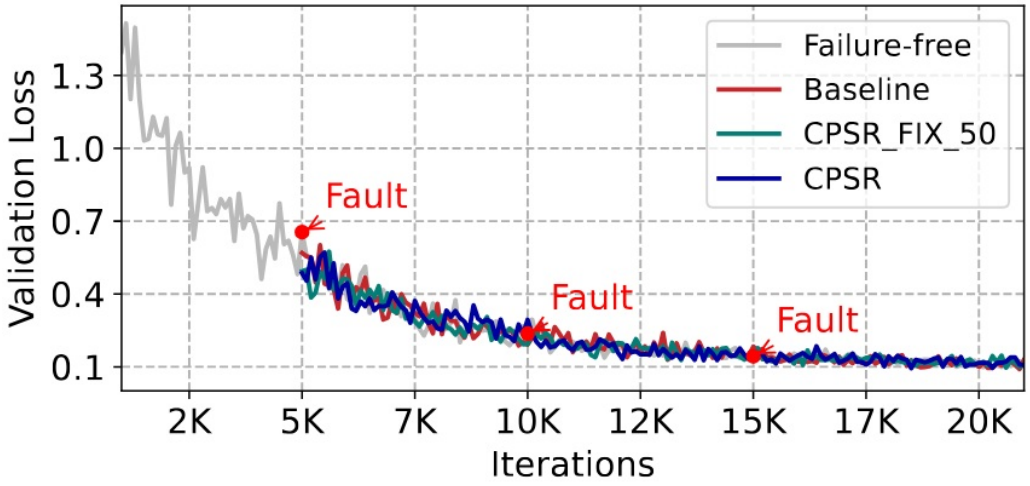
Model	Dataset	# Layer	# Hidden size	# of Params
VGG16	ImageNet	16	_	144M
GPT2-0.35B	CodeSearch	24	1024	350M
GPT3-3.35B	CodeSearch	16	4096	3.35B



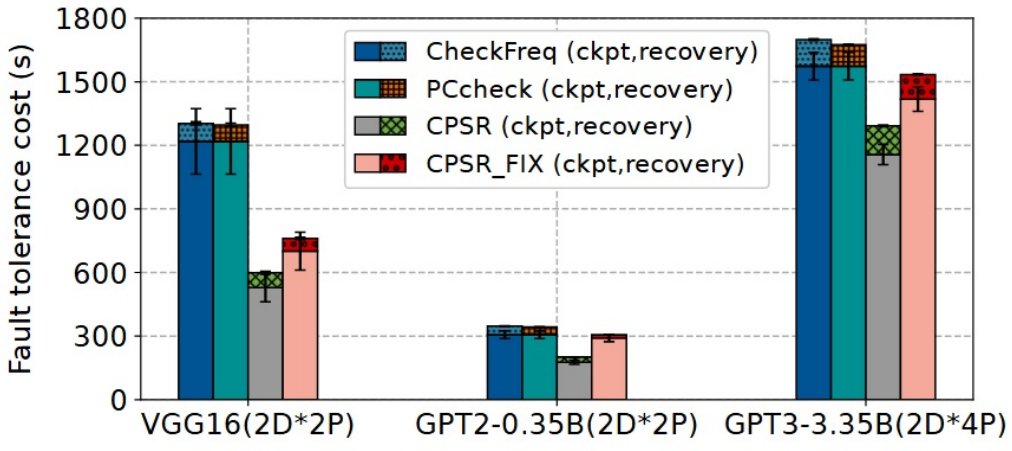
(a) Failure at 25% training progress. (b) Failure at 50% training progress. (c) Failure at 75% training progress.

- CPSR reduces the **recovery cost** by **41.66%** on average compared with state-of-the-art approaches.

Evaluations (overall fault-tolerant cost)



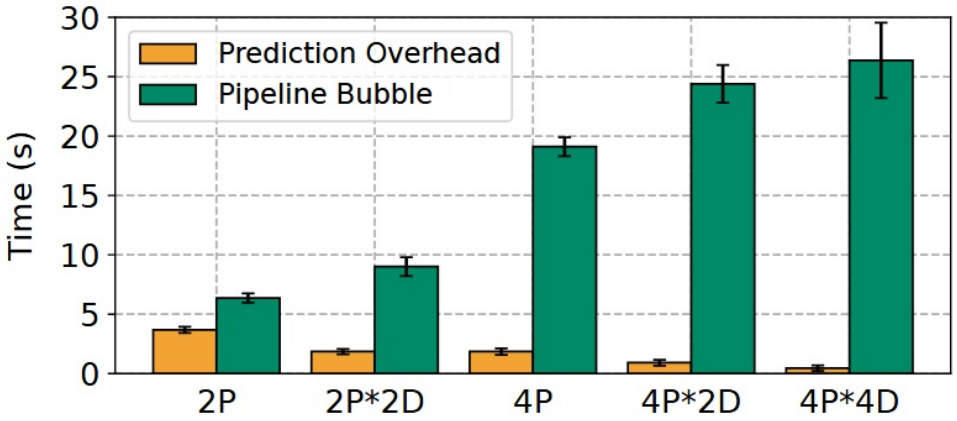
Validation loss curves.



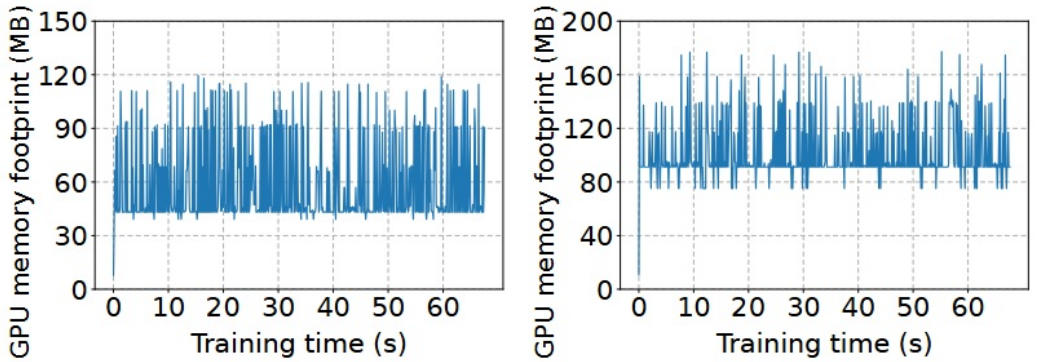
Fault-tolerant cost comparison.

- CPSR achieves the smallest fault-tolerant cost while staying the closest to the failure-free training trajectory.

Evaluations (overhead)



Predictor overhead.



(a) Token size of 1024 (GPT2-0.35B). (b) Token size of 4096 (GPT3-3.35B).

GPU memory footprints

- CPSR is suitable for **larger-scale training scenarios**.

Conclusion and Future Work

Conclusion

i. A predicting-based recovery method is proposed to provide controlled fast recovery of DNN trainings without recomputation.

ii. A quantified recovery cost model and a novel checkpoint interval problem is introduced. A solution is proposed to compute the optimal checkpoint intervals.

iii. Extensive testbed experiments are conducted to evaluate the efficiency of CPSR.

Future Work

i. Better prediction model.

ii. Better training data preparation method for larger models.

Temple University

Thank you

Presenter: Hongliang Li

lihongliang@jlu.edu.cn
College of Computer Science and Technology,
Jilin University, Changchun, China, 130012