Hello!

CS 744: JUST-IN-TIME CHECKPOINTING

Shivaram Venkataraman

Spring 2025

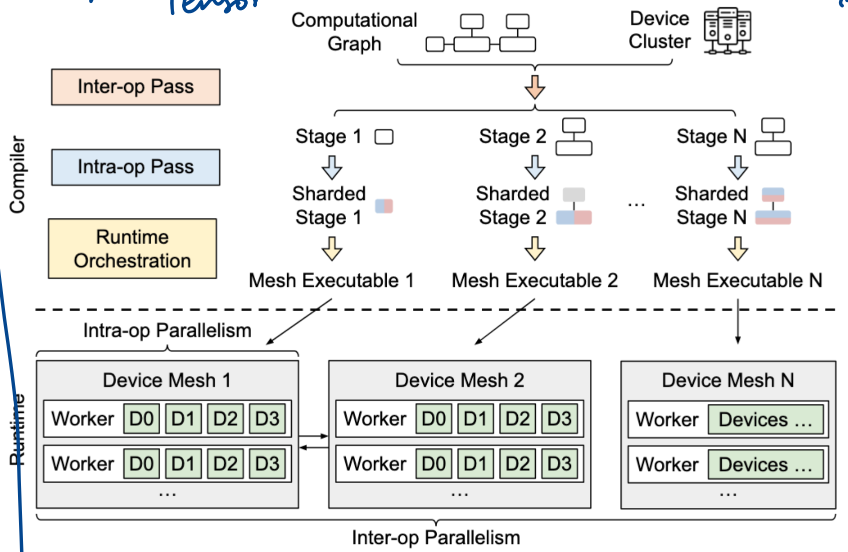
ADMINISTRIVIA

Assignment 2 is due TODAY

Project Preference form due Monday (17^{th})

3D PARALLELISM, ALPA

- Data
- Model (Pipeline)
- Tensor



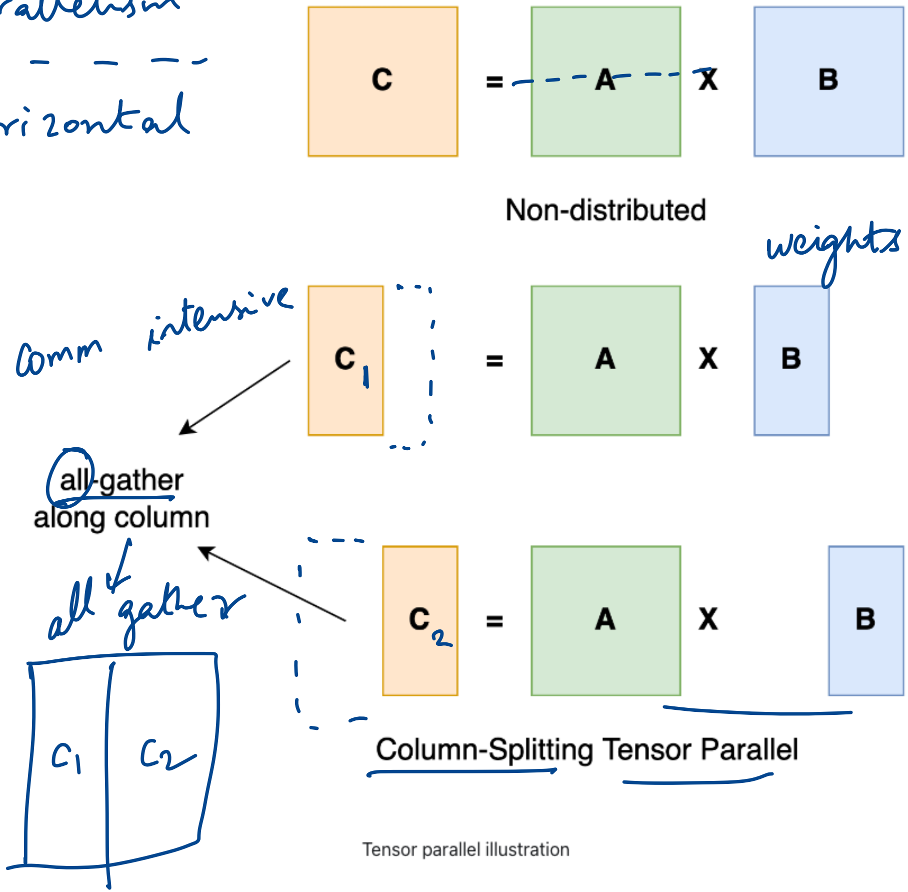
8 GPUs

= 2 × 2 × 2 = 8 × 1 × 1

DP PP TP

Data parallelism

horizontal



Tensor parallel illustration

ERROR FREQUENCIES

memory errors

Error type	F	N
ECC errors	16	31
NCCL errors	12	33
CUDA errors	9	9
GPU lost errors	15	17
infoROM errors	9	19
Other GPU failures	7	10
IB errors	6	10
Software bugs	9	9
Other	16	17

infiband errors

OPT 175B Training Error Counts

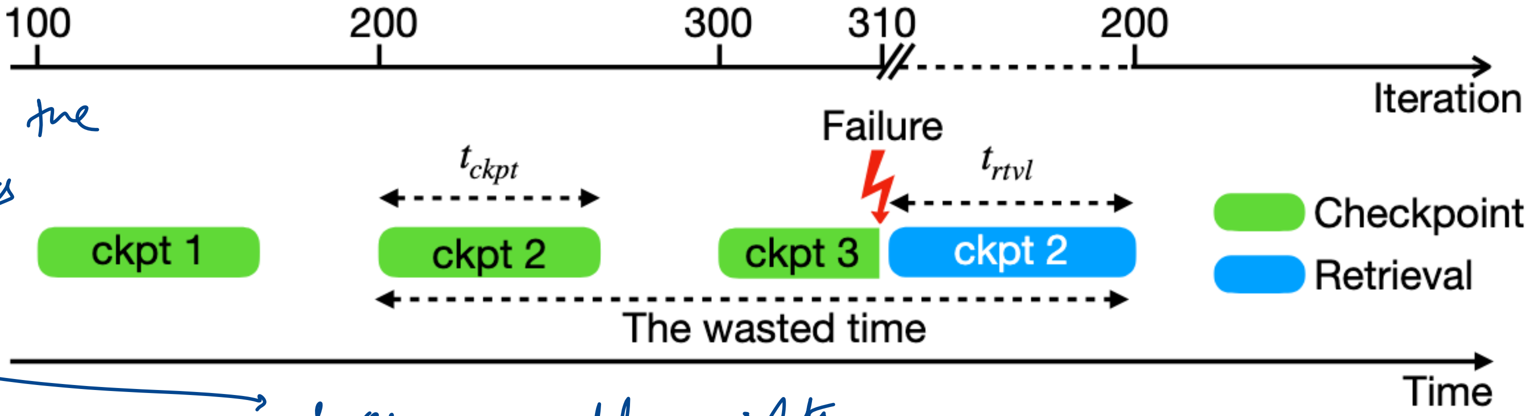
https://github.com/facebookresearch/metaseq/blob/main/projects/OPT/chronicles/OPT175B_Logbook.pdf

larger models → more GPUs
more machines

larger datasets → longer training
≃ months

CHECK-POINTING BASED FAULT TOLERANCE

one of the workers



saves model weights optimize state etc. to some persistent storage (HDFS)

Downsides?

↳ wasted work

↳ overhead of ckpt

APPROACH

combine with periodic checkpointing

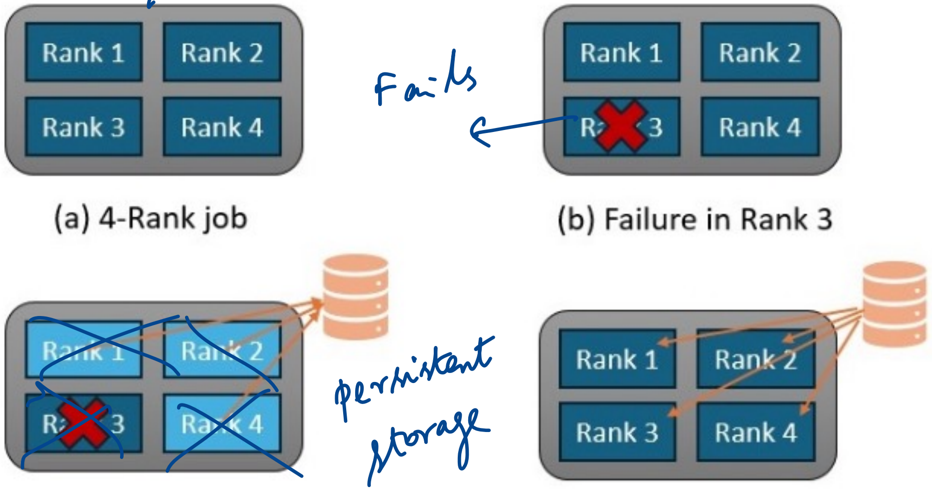
Take checkpoints after failure!?

no checkpoints

Just-in-time

redo

↳ one minibatch worth of work!



(a) 4-Rank job

(b) Failure in Rank 3

(c) Other ranks hang and checkpoint GPU state

(d) Ranks restored from saved checkpoint

Limitation: Replication in the parallelism strategy

CHALLENGES?

How do we know when a failure happens?

→ anytime replicas → all_reduce calls

How do we get access to GPU state?

User-level → save_ckpt

Transparent → user code is unmodified

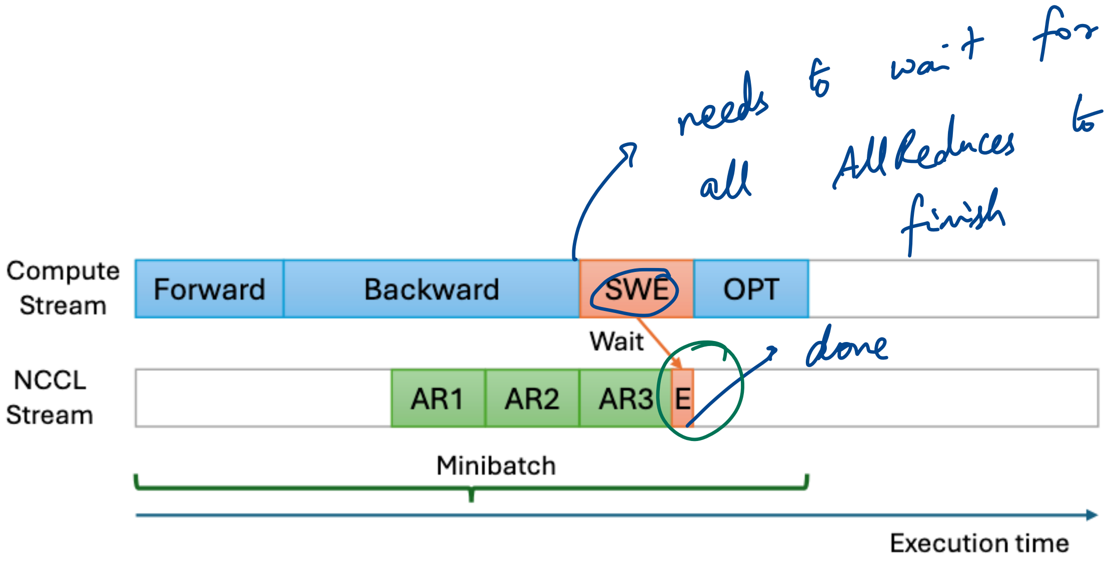
USER-LEVEL JIT CHECKPOINTING

1. In each rank, detect hangs during AllReduce

    Timeout cudaEvent

    ↳ mark the AR as complete

    watch dog thread

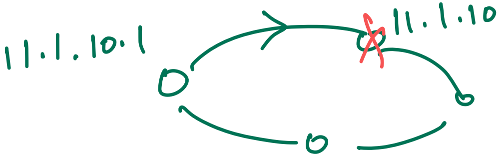


2. Healthy ranks checkpoint state
3. Restart the job
4. Load the relevant checkpoint

USER-LEVEL JIT CHECKPOINTING

ring reduce

gloo / nccl



1. In each rank, detect hangs during AllReduce

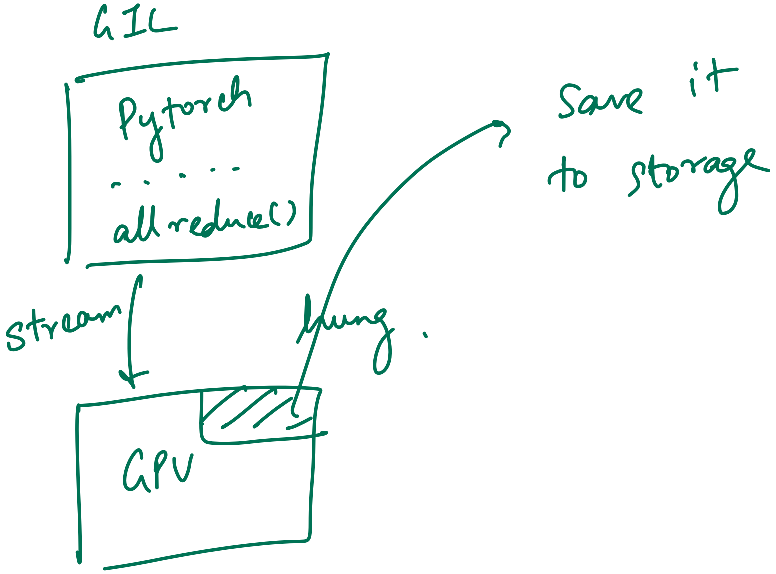
2. Healthy ranks checkpoint state

   Release GIL, Memcpy stream

3. Restart the job → restart all of the ranks

4. Load the relevant checkpoint → resume training from the checkpoint

   Iteration i vs. i + 1



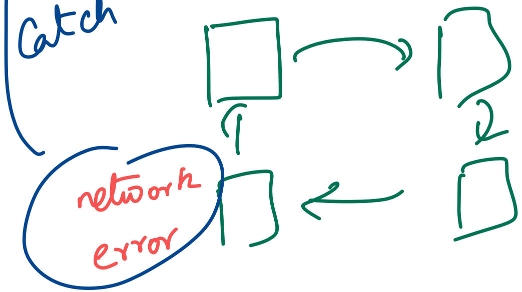
TRANSPARENT JIT CHECKPOINT

No changes to user code

Goal: Prevent errors from crashing PyTorch process

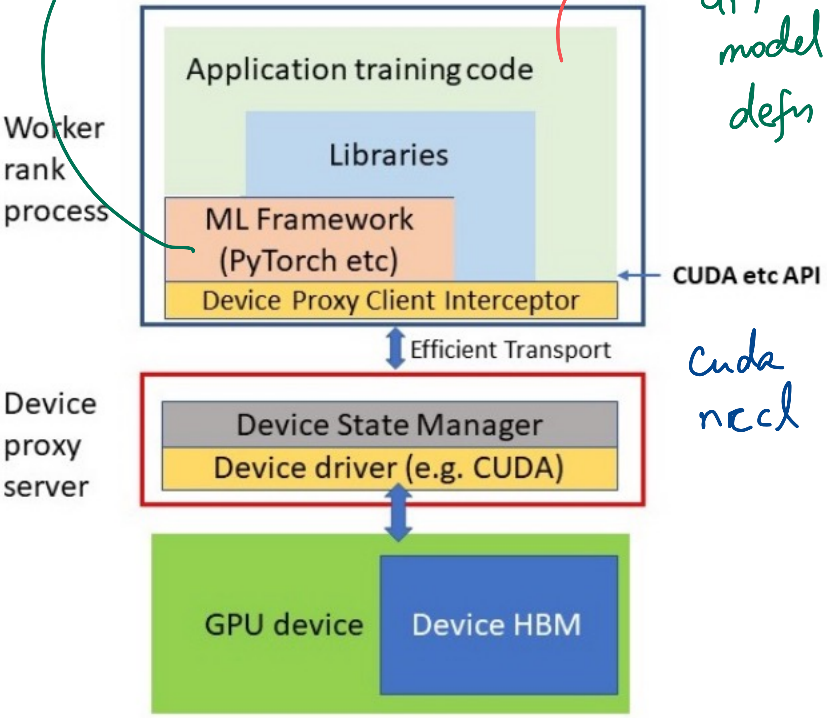
Build a proxy server

catch



not user code ?

train.py
Resnet
GPT model defn



cuda
nccl

TRANSPARENT JIT CHECKPOINT

Steady State Work

Log all CUDA / NCCL APIs

Replay log → Redo logging

Clear log every mini-batch

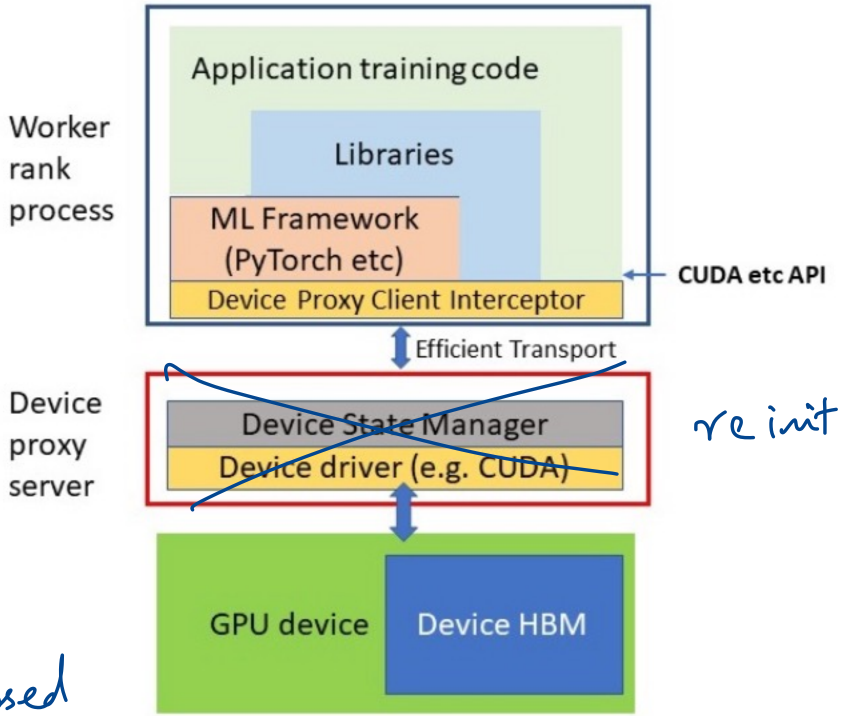
hooks in PyTorch

↳ nccl AllReduce calls

Validate the log periodically

↳ replay

Compare directly passed thru calls



re init

TRANSPARENT JIT CHECKPOINT

not fatal

Recovery Work – GPU/Network error

Catch error in the Device API

PyTorch/ML framework unaware!

Goal: restart device proxy server

Need to re-fill GPU state

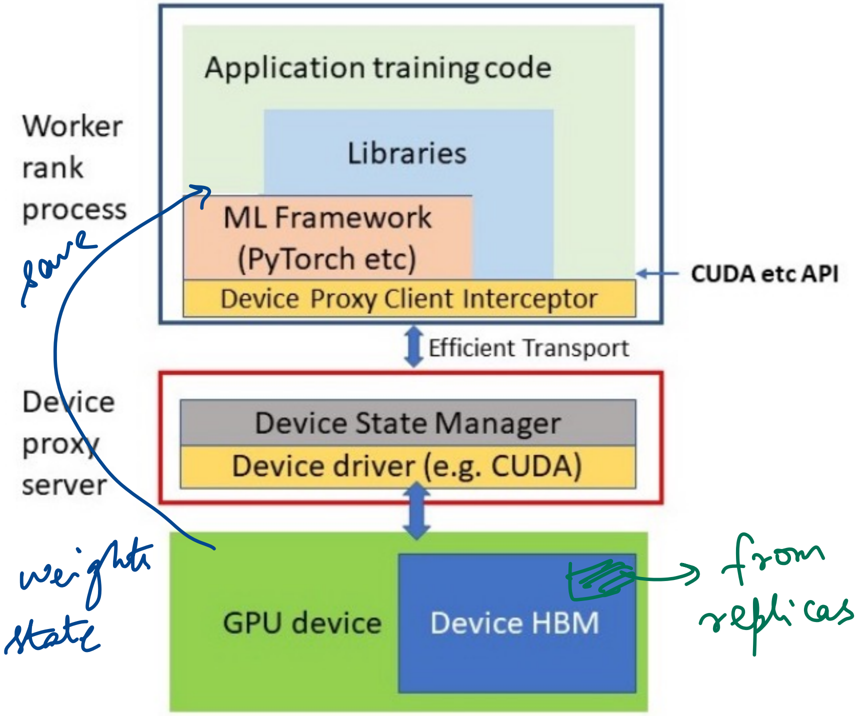
from CPU or replicas

model weights

opt state

etc.

save



Iteration i vs. i + 1

RECOVERING FROM GPU FAILURES

Take a consistent checkpoint of CPU state (all processes!)

CRIU on Linux

→ checkpointing library → file

VM migration

Restore GPU buffers from checkpoint (of other ranks)

→ blank GPU initialization

replay the log → all reduce call

state from other ranks

move to a diff without user code knowing about it!

SUMMARY

Checkpointing based approach for DNN fault tolerance

Mitigate overheads with just-in-time checkpoints

User-based and transparent approaches

DISCUSSION

https://forms.gle/QZh1zRC1TVbugwAx7

Model	#Params(B)	#GPUs	Parallelism	Framework
GPT2-S	0.124B	4xA100	4D	Megatron-DS
GPT2-S-3D	0.124B	8xV100	2D-2P-2T	Megatron-DS
GPT2-XL	1.5B	8xV100	2D-2P-2T	Megatron-DS
GPT2-8B	8.3B	2x(8xV100)	2D-4P-2T	Megatron-DS
GPT2-18B	18B	4x(8xV100)	2D-4P-4T	Megatron-DS
BERT-L-PT	0.334B	8xV100	8D	Megatron
BERT-B-FT	0.110B	8xV100	8D	Hugging Face
T5-3B	3B	2x(4xA100)	FSDP	PyTorch
ViT	0.632	8xV100	8D	PyTorch
PyramidNet	0.24B	4xA100	4D	PyTorch

Table 2. Experimental workloads used for evaluation. PT=Pre-training, FT=Fine-tuning, DS=DeepSpeed. For parallelism, 2D-4P-2T means 2-way Data-parallel, 4-way Pipeline-parallel, 2-way Tensor-parallel. For GPUs, 2x(8xV100) means 2 nodes of 8 V100s each.

Model	PC_disk	PC_mem	CheckFreq	PC_1/day	JIT-C
GPT2-S	0.042	0.042	0.024	0.004	0.0024
GPT2-XL	0.093	0.078	0.047	0.007	0
GPT2-8B	0.216	0.186	0.111	0.02	0
GPT2-18B	0.330	0.275	0.166	0.02	0
BERT-L-PT	0.07	0.068	0.031	0.005	0.0076
BERT-B-FT	0.039	0.036	0.026	0.0016	0

Table 3. Checkpointing overhead percentages for Periodic Checkpointing baselines, assuming optimal checkpointing frequency, compared to JIT-Checkpointing (JIT-C)

encoder / decoder

limited savings?

disk ~ mem
not flushing ??

Bigger models suffer
Periodic ckpts

Project ideas / Anybody looking for projects?

→ Flink → Can you combine this idea?

→ Spark - AQE → parameter tuning

↓

automate ?

tuning

NEXT STEPS

Next class: LLM Inference

Project Preference form