Towards a Flexible, Efficient, and Resilient AI Training on AMD GPUs with DeepSpeed Universal Checkpointing

Pratik Mishra
AMD

August 6, 2025
FMS: Future of Memory and Storage 2025

AMD
together we advance_

Agenda

- AI Systems Glossary 101s
- Infrastructure, Reliability, and Foundation Model Training
- Fault-Tolerance Tax
- Universal Checkpointing (UCP) : *Collaboration with UIUC (Prof. Minjia Zhang)*
- Conclusion
- Copyrights and Disclaimer

SDC'24

AI Training Infra Reliability 101: Metrics

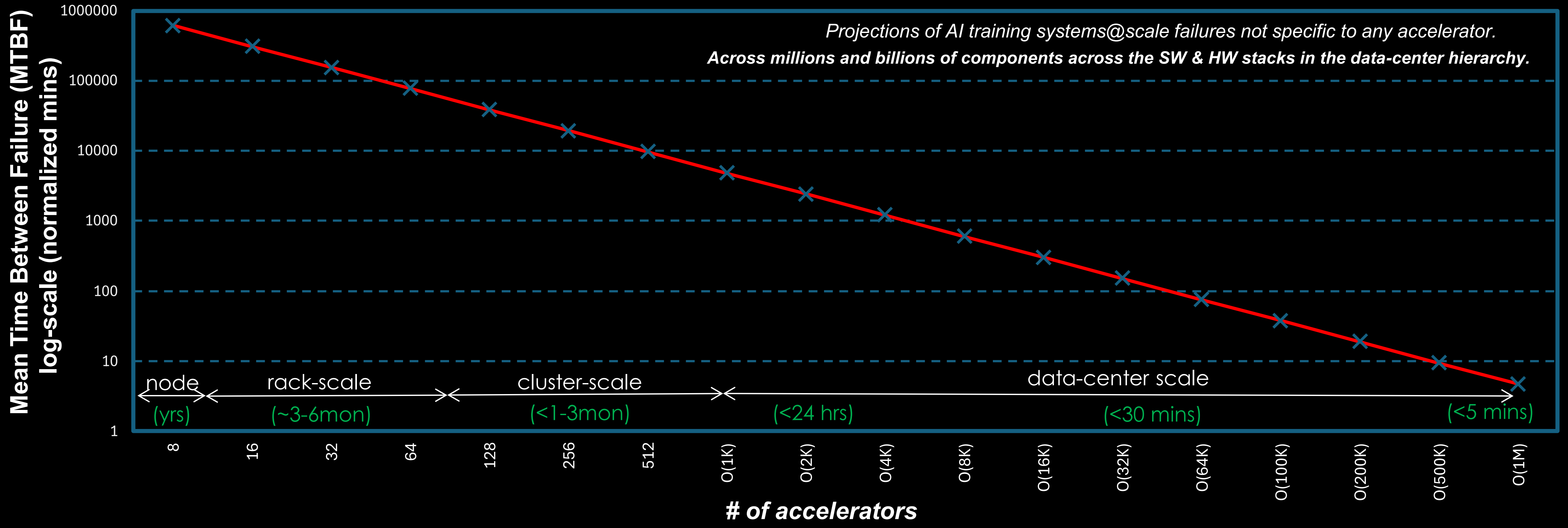
- **Training Goodput** = Actual progress made / total time
- **<u>M</u>odel <u>F</u>LOPs <u>U</u>tilization (MFU)** = FLOPs a model utilizes/ peak HW FLOPs available.
- **<u>M</u>ean <u>T</u>ime <u>B</u>etween <u>F</u>ailures (MTBF)** = total time / # of failures.
- **<u>E</u>ffective <u>T</u>raining <u>T</u>ime <u>R</u>atio (ETTR)** = actual training time / total time

Achieving high training goodput and maximizing model FLOPs utilization to improve the Effective Training Time Ratio remains a significant and ongoing challenge.

Failures and Training Efficiency?

Reliability and Training Efficiency @scale

$$MTBF \propto 1/(no.of\ accelerators)$$



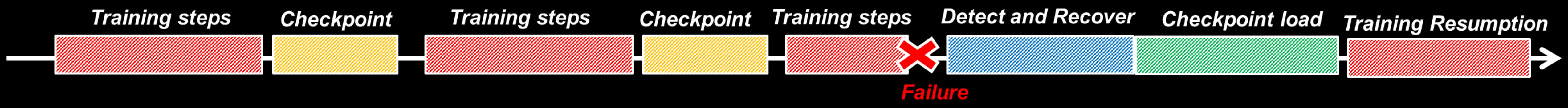
With growing scale of AI deployments, the MTBF decreases significantly.

Therefore, resiliency is the core for achieving Training efficiency and increasing Training Goodput and ETTR.

Fault Tolerance, Training Efficiency and Checkpointing

- Fault-tolerance, resiliency, and recovery are of utmost importance for Training Efficiency metrics (discussed earlier).

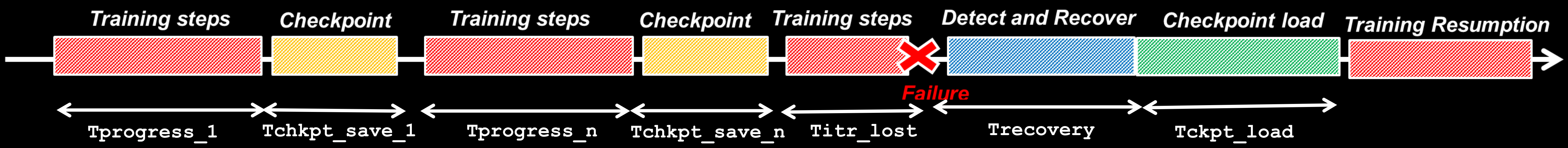
- Storage community's poster AI use-case: ***Checkpointing***.



- Critical fault-tolerance mechanism for periodically persisting training snapshots to enable recovery via rollbacks in the event of failure.
  - *Also: Hardware refresh, Resource re-balancing, post-training, concurrent evaluation, increase accuracy, etc.*

With scale and every-lowering MTBFs, the checkpointing frequency, size, and complexity increases significantly; imposing heavy data-center tax (GPU underutilization).

Fault Tolerance Tax: Checkpointing



$$FT_{overhead} = T_{chkpt\_save} + T_{itr\_lost} + T_{recovery} + T_{chkpt\_load}$$

$$ETTR = (1 - FT_{overhead})$$

- Achieving optimal ETTR @ data-center scale is **"real"** challenge.
  - *Without optimization, systems may spend more time managing failures than actual training.*
  - **Trade-off**: Excessive checkpoints increases data-center tax & infrequent increases risks (cost).
  - **Data-center tax**: compute, network, storage.

Therefore, to achieve optimal* ETTR (+goodput) *it is quintessential for reliability mechanisms to strike the balance of performance, scalability, and cost-effectiveness.

Optimizations : Checkpointing

$T_{\texttt{chkpt\_save}}$

- **<u>S</u>**erialization + **<u>P</u>**ersistence → {GPU states + CPU states + metadata}
- *Synchronous chkpt*: simple but introduces significant training stalls.
- *Asynchronous checkpoints* (PyTorch DCP) reduces persistence latency (lesser ETTR) by alleviating main GPU thread from critical IO path.
- *Needs optimizations* to reduce@scale overheads (BW, //sm, etc.)

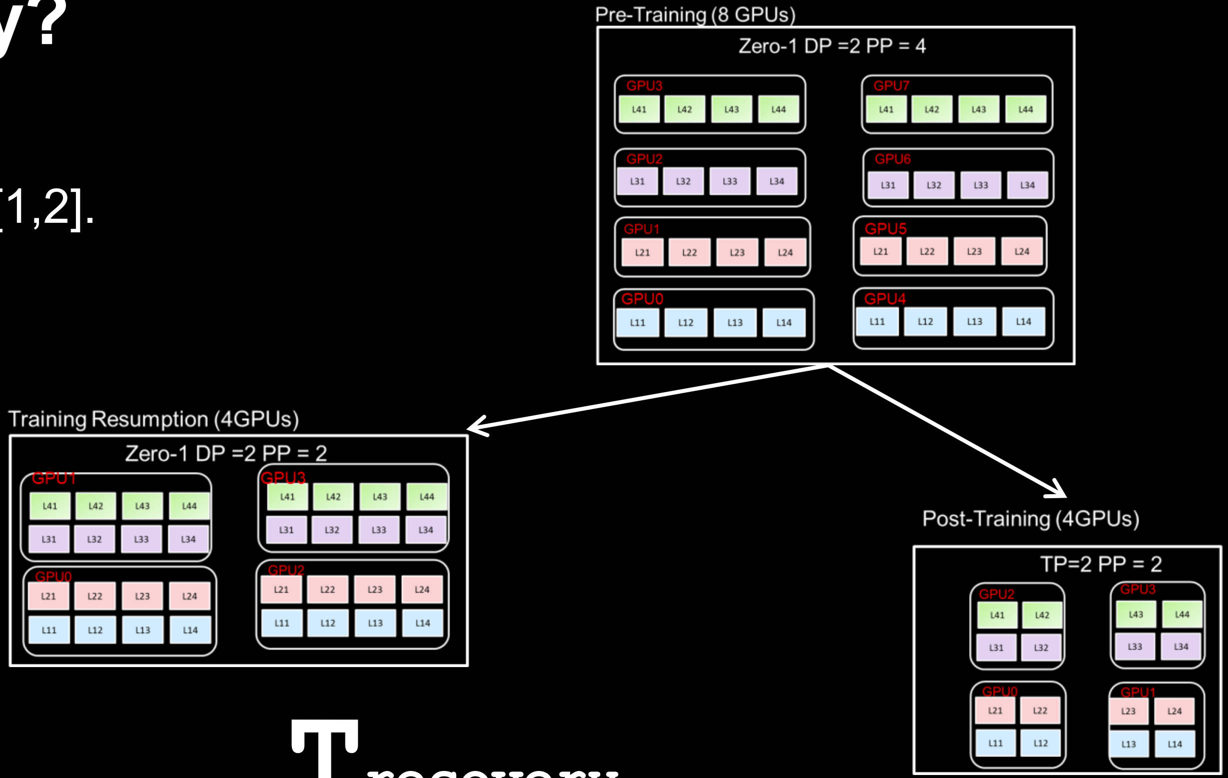
$T_{\texttt{chkpt\_load}}$

- Loading checkpoint is **mission-critical.**
- **<u>L</u>**oading + **<u>D</u>**eserialization: *impacts training resumption (ETTR, MFU)*.
  - Also, post-training and inference.
- Concurrent loading (size, magnitude) can destabilize infrastructure.
- GPU node BW, Frontend network BW, storage throughput, cluster topology, **reconfiguration**, etc.

Efficient fault-tolerant checkpointing at scale requires GPU–storage path optimizations and topology-aware strategies to sustain robust infrastructure and high MFU.

Recovery with Flexibility + Elasticity?

- Resource rebalancing (GPU shape change) is common [1,2].
  - **Training Resumption :** reconfiguration parallelism.
  - **Post-Training** : lower requirement for SFT, RL.
  - **Inference** : much lower with diff. config + data-set.



- Existing distributed training frameworks provide highly limited support for reconfiguring //sm.
  - Mostly inefficient: offline, hand-written scripts, human intervention.

$T_{recovery}$

$+ \quad T_{chkpt\_load}$

Distributed checkpoints are tightly coupled to initial parallelism and HW configuration, resulting in GPU idle time (recovery time) during re-sharding limiting adaptability to resource elasticity.

[1] Bytedance 2024
[2] DeepSpeed UCP 2024

Supporting flexible, efficient and resilient training on AMD GPUs with DeepSpeed Universal Checkpointing

AMD
together we advance_

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Collaboration: Prof. Minjia Zhang (UIUC), co-creator of DeepSpeed UCP; and PhD students (Jiankun Wang and Xinyu Lian)

DeepSpeed UCP

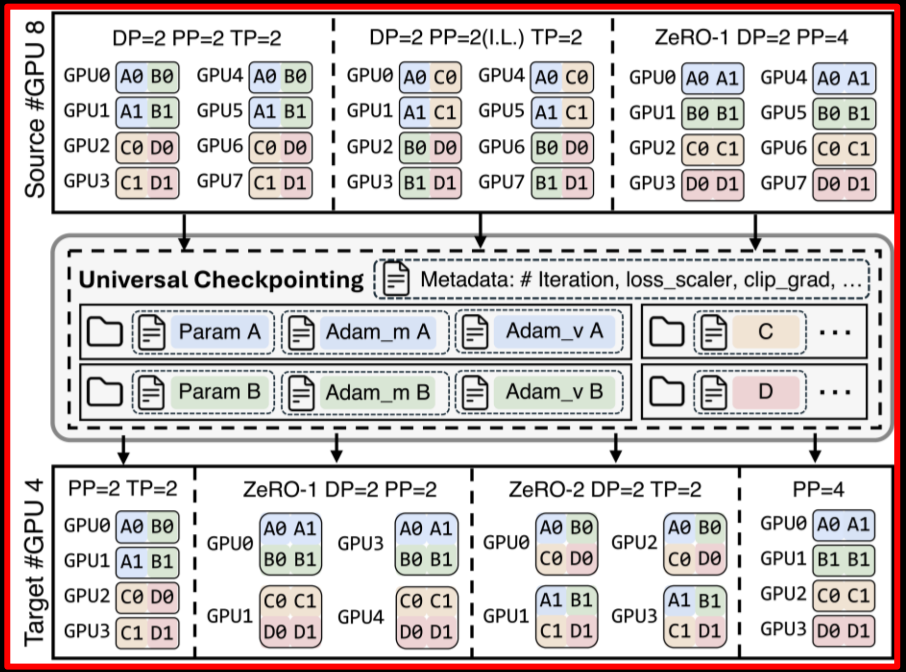
[1] Github: https://github.com/deepspeedai/DeepSpeed/blob/master/blogs/deepspeed-ucp/README.md

[2] Paper: Lian, Xinyu, et al. "Universal checkpointing: Efficient and flexible checkpointing for large scale distributed training." *arXiv preprint arXiv:2406.18820* (2024). ***Accepted in USENIX ATC'25.***

UCP : Universal Checkpointing

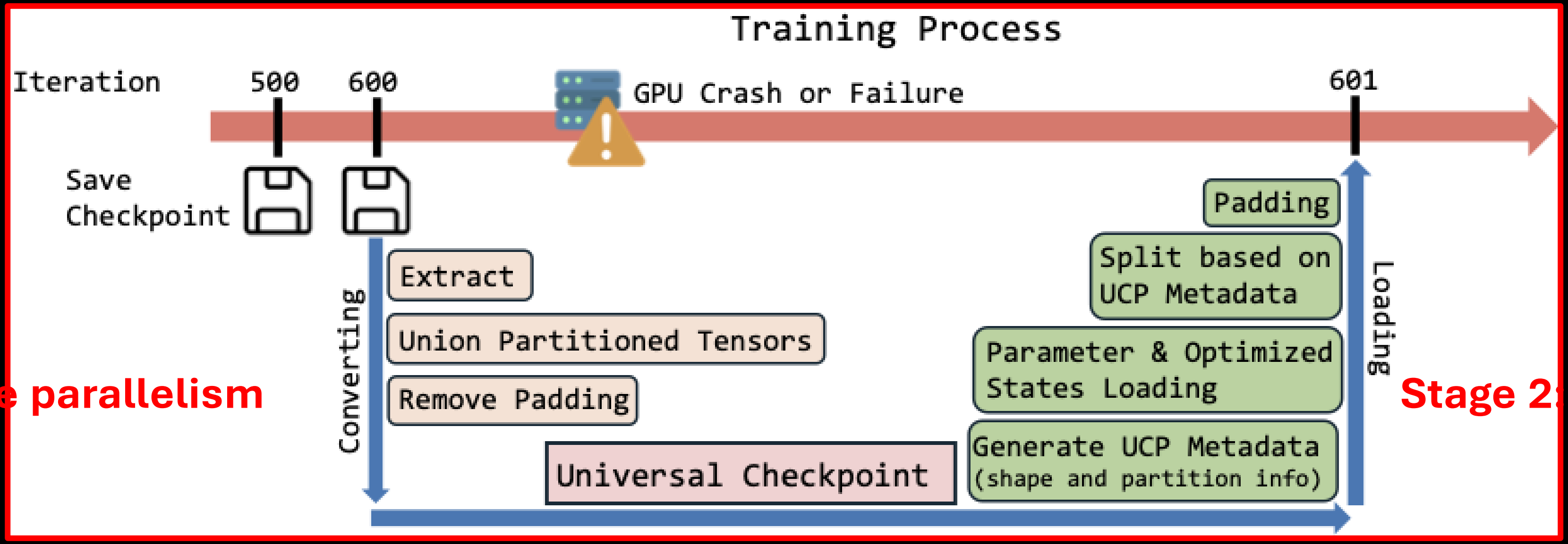
- Developed as a part of DeepSpeed.
  - Support for commercial-scale models (BLOOM, Megatron GPT, Llama, Microsoft Phi)
- Comprehensive, Flexible, and automated.
  - Checkpoint re-sharding along most training parallelism techniques
    - Combinations - Zero-DP, PP, TP, DP, SP.
  - Defines UCP language to support checkpoints from various frameworks (for e.g. DCP)
    - Pattern matching: runtime-sharding information.

DeepSpeed UCP 2024



UCP : 100K birds-eye view

DeepSpeed UCP 2024



Stage 1: Decouple parallelism

Stage 2: Load and Convert

From source distributed checkpoints re-create per-parameter consolidated view/ “atomic checkpoints.”

“atomic checkpoints” per parameter: Weight, Momentum, Variance.

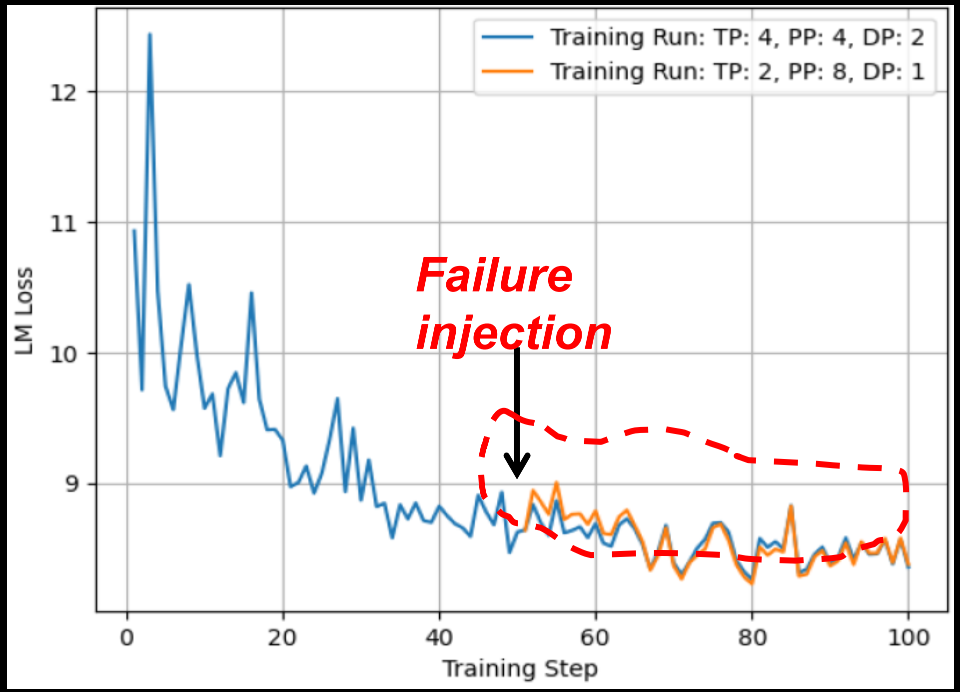
Based on UCP language pattern-matching; re-shard from atomic checkpoints to target GPU configurations.

UCP: Accuracy

Recovery from checkpoint needs to be accurate, fast and agnostic to changing parallelism patterns.

Blue denotes the actual training run loss, and orange denotes the loss after checkpoint recovery with changing parallelism.

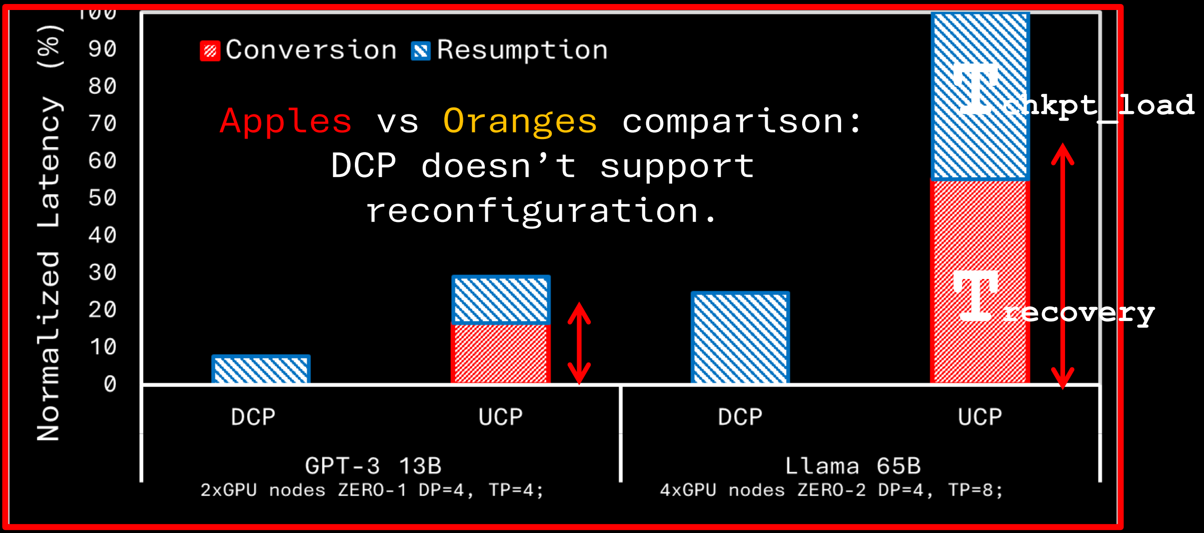
Experiments over GPU clusters and remote high-performance NVMe storage system.



Initial Configuration (blue) : 8 MI210 nodes (32 GPUs) with TP=4, PP=4, DP=2.
Resume Training (orange) : 4 MI210 nodes (16 GPUs) with TP=2, PP=8, DP=1.

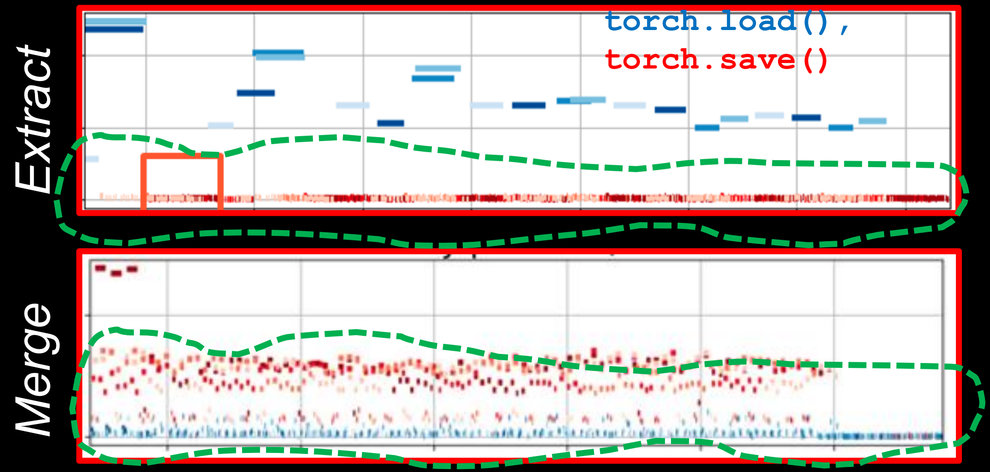
UCP enables failure recovery with resource rebalancing (GPU shape, parallelism) without compromising training accuracy.

UCP: Under the Hood Analysis



- **UCP** has to do ***extra work*** compared to DCP for reconfiguration:
  1) Decouple parallelism 2) Convert and Load to target GPU shapes.

UCP IO volume > 4x DCP due to reconfiguration.



Access pattern – Decouple parallelism

- High GPU node host-resource consumption:
  - Large # of temporary intermediate files.
  - Time, size and phase-varying access pattern.
- GPU – remote storage BW underutilization:
  - Serialization and Deseralization
  - Opportunity to exploit In-node parallelism.

UCP needs to perform extra work for elastic recovery.
However, it needs adaptive optimizations to reduce recovery time cost-effectively.

UCP: Architectural Re-design

Infrastructure-aware optimizations

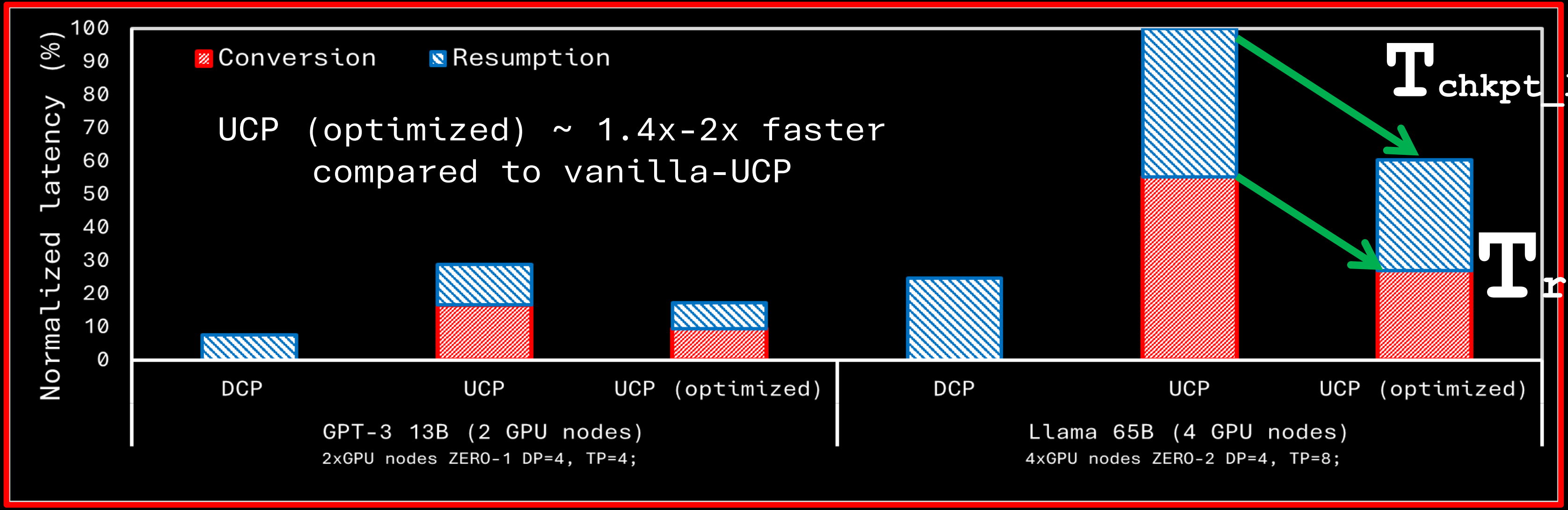
- Storage characteristic - throughput, backend (object/file), scalability analysis, etc.
- Cluster and GPU-node topology (network BW).

+ Inter/intra-node optimizations

- Dynamic, adaptive GPU-node host-resource (memory, compute) + workload-aware.
- Multi-node + async Hierarchical parallelism.

+ Metadata-aware optimizations

- Deserialization chkpt (.pt) file structure-aware.
- mmap + offset-based dynamic loading : *Elimination of temporary file creation*.



T_{chkpt_load} ~1.5x faster

IO traffic 2x reduction.

$T_{recovery}$ ~2x faster

Reduction in GPU-node host resource consumption, IO traffic, etc.
Increase in GPU-Storage BW utilization.
Resulting in lower latency.

UCP optimizations across the GPU–storage data path significantly reduce recovery and resumption time (+cost), improving training goodput and lowering ETTR.

Conclusion

- ***Trend is clear***:
  - With scale and size of AI deployments, failures will be inevitable, while MTBF will keep lowering.
  - Robust, scalable, and cost-effective fault-tolerance recovery and resiliency mechanisms is the core to achieve optimal ETTR and Training goodput.
  - Resource rebalanced recovery is becoming common in the AI lifecycle.
  - Therefore*, AI Training Fault-tolerance needs to be flexible, resilient, elastic and adaptable.*

- UCP (Universal Checkpointing) seems to be promising direction for automated, flexible, resilient, and elastic AI Training. *However, it needs full-stack scalable optimizations.*

Therefore, to achieve optimal* ETTR (+goodput) *it is quintessential for reliability and recovery mechanisms to strike the balance of performance, scalability, and cost-effectiveness to harness the full potential of GPU-accelerated AI computing.

COPYRIGHT AND DISCLAIMER

AMD