KubeCon CloudNativeCon Europe 2026

Masaryk University | TÉCNICO LISBOA | UNIVERSITY OF OXFORD

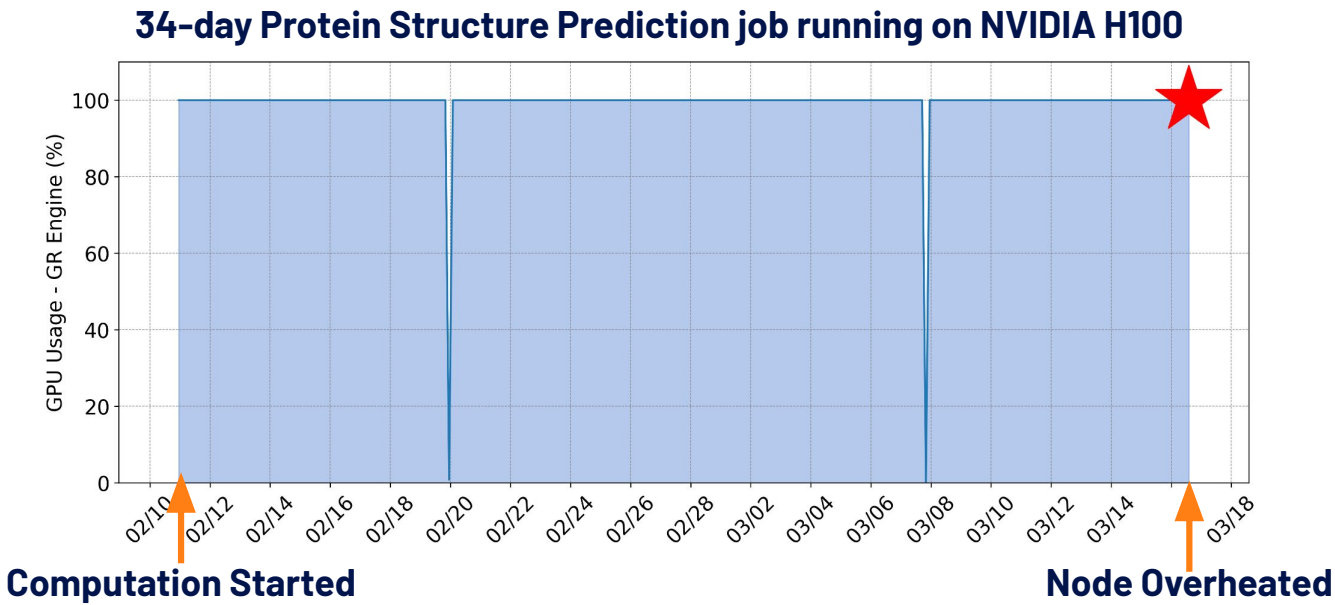
Optimizing Error Recovery for Cost-Efficient Distributed AI Model Training with Kubernetes

Viktória Spišaková, Radostin Stoyanov, Andrey Velichkevich
Collaboration with Adrian Reber, Peter Hunt
Supervisors: Prof. Rodrigo Bruno, Prof. Wes Armour

Red Hat | DevZero

Real-World Experience

Managed Kubernetes cluster at e-INFRA CZ



Sudden Node Failure ⇒ Job Stuck ⇒ Node Reboot
No results ⇒ €1,207 in compute costs wasted

Challenges with Distributed Workloads

- Require gang-scheduling if tightly-coupled (*all-or-nothing* placement)
- Workload failures happen due to various reasons [1,2,3]
- One eviction cascades across the entire job
- A distributed job is only as reliable as a least reliable infrastructure component

[1] Revisiting Reliability in Large-Scale Machine Learning Research Clusters. IEEE HPCA (2025)
[2] Just-In-Time Checkpointing: Low Cost Error Recovery from Deep Learning Training Failures. EuroSys (2024)
[3] Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads. USENIX ATC (2019)

Distributed Workload Failures

- Infrastructure
  - Network issues, slow filesystem mounts, NVLink/PCI/IB errors
  - Infrequent but long runtime-to-failure (MPI runtime errors)
- AI Engine: CUDA failure (init, versions mismatch), CPU/GPU OOM
- User: CPU OOM, permission errors, misconfigurations, wrong semantics

[1] Revisiting Reliability in Large-Scale Machine Learning Research Clusters. IEEE HPCA (2025)
[2] Just-In-Time Checkpointing: Low Cost Error Recovery from Deep Learning Training Failures. EuroSys (2024)
[3] Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads. USENIX ATC (2019)

Challenges with Distributed Training

GPU failures require restarting training jobs

- 54 days training: 466 job interruptions[1]
  - ~78% of unexpected interruptions due to hardware errors
- 3-23 hours MTBF on older GPUs[2]
- Estimated monthly cost up to a few million dollars[3]

[1] The Llama 3 Herd of Models. arXiv:2407.21783 (2024)
[2] Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads. USENIX ATC (2019)
[3] Just-In-Time Checkpointing: Low Cost Error Recovery from Deep Learning Training Failures. EuroSys (2024)

Failure Recovery Methods

Model Checkpoints: Error recovery in user code

- Requires restarting training jobs & re-running application code
- Involves (potentially) large initialization overheads
- Frequent checkpointing of large models leads to long GPU idle times

Infrastructure Checkpoints: Transparent system-level error recovery [1,2,3]

- Enables checkpointing of jobs without any user code changes
- Supports transparent job migration with existing schedulers

[1] CRIUgpu: Transparent Checkpointing of GPU-Accelerated Workloads. arXiv:2502.16631 (2025)
[2] *Just-In-Time Checkpointing: Low Cost Error Recovery from Deep Learning Training Failures*. EuroSys (2024)
[3] *Singularity: Planet-Scale, Preemptive and Elastic Scheduling of AI Workloads*. arXiv:2202.07848 (2022)

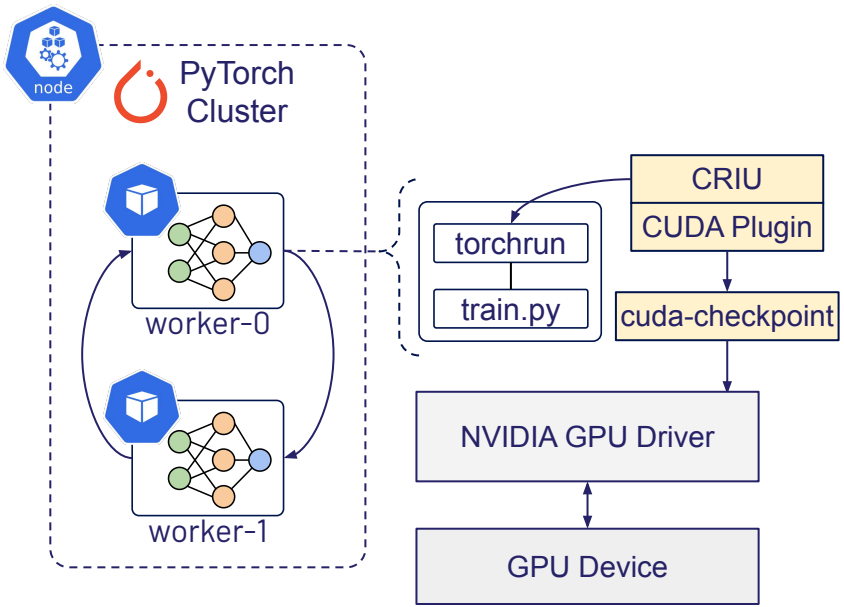
Transparent Checkpointing of GPU Workloads

Enabling transparent fault-tolerance of training jobs

Transparent Checkpointing of AI/ML Workloads in Kubernetes. KubeCon Europe 2025

Coordinated Checkpointing for Distributed HPC Applications. KubeCon Europe 2024

Transparent GPU Checkpoint/Restore

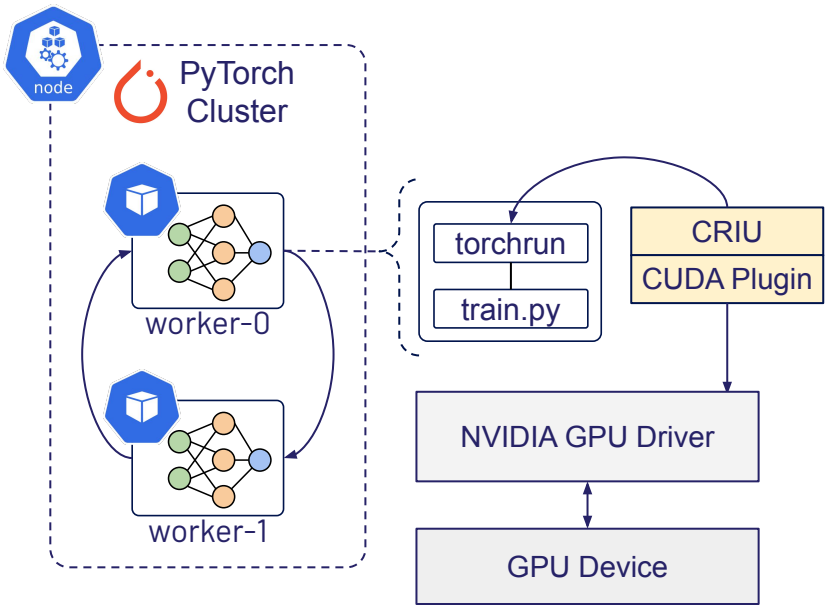


CUDA Checkpoint API Actions[1]

- **Lock**
  - Blocks CUDA driver APIs from launching work or modifying GPU state for the target process
- **Checkpoint**
  - Completes pending GPU work
  - Copies VRAM to driver-managed host allocations
  - Releases GPU resources
- **Restore**
  - Copies host allocations back to VRAM
  - Restores memory mappings
- **Unlock**
  - Unblocks CUDA driver APIs, allowing to resume

[1] CUDA Driver API :: CUDA Checkpointing

Transparent GPU Checkpoint/Restore



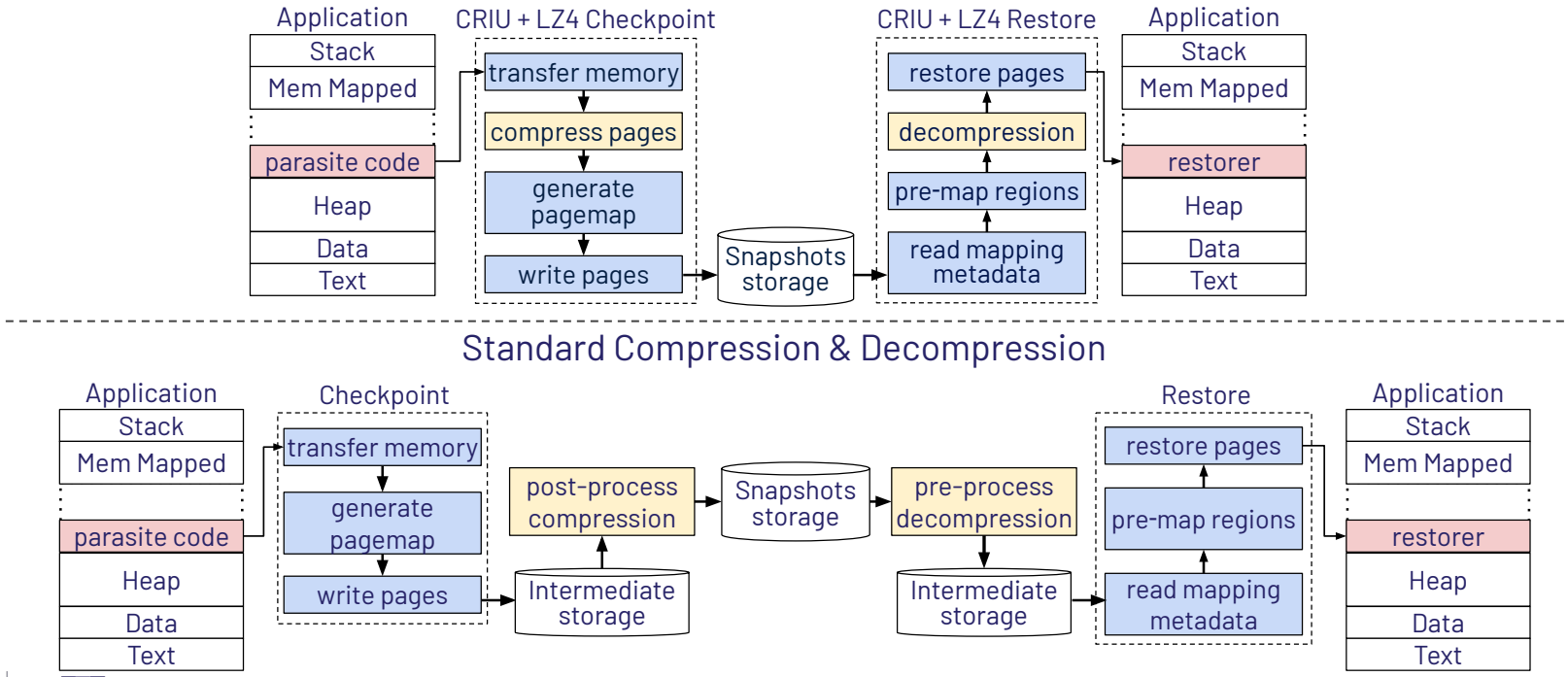
Self-contained CUDA Plugin

- Direct CUDA Driver API Integration
- Simplified installation on Kubernetes nodes (no separate helper binary)
- Eliminates (fork+exec+waitpid) overheads

Optimization Techniques for Checkpointing

Enabling low-overhead checkpointing

On-the-Fly Memory Compression



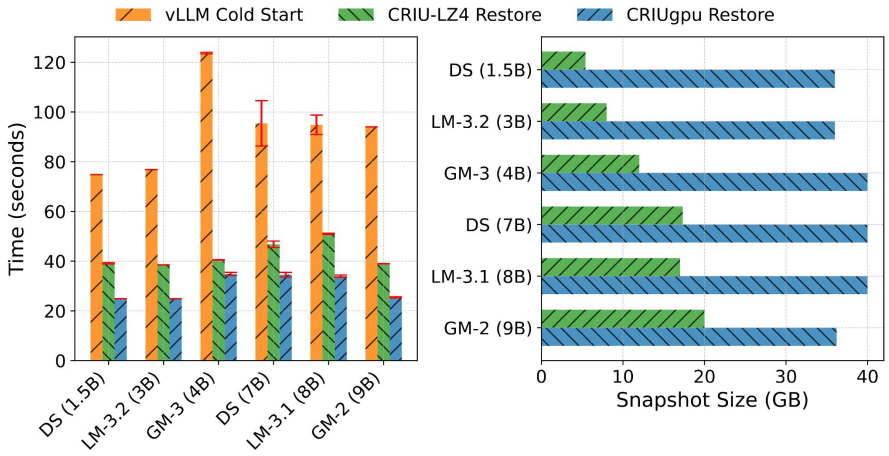
On-the-Fly Memory Compression

Virtual Address	PyTorch Allocation	LZ4 Compressed	Stored As
`0x7f3ce4a4a`	nn.Linear.weight (frozen 4-bit base model weights)	1.1 KB	compressed
`0x7f3ce8200`	LoRA adapter B (trained float16 adapter weights)	3.9 KB (incompressible)	raw (copy)
`0x7f3cf0000`	Optimizer.state (AdamW exp_avg, near-zero values)	2.6 KB	compressed
`0x7f3d10000`	gradient buffer (loss.backward() output)	3.8 KB (high entropy)	raw (copy)
`0x7f3d20000`	CUDA context (unused region)	0 bytes	zero marker

Compression Handling by Memory Type

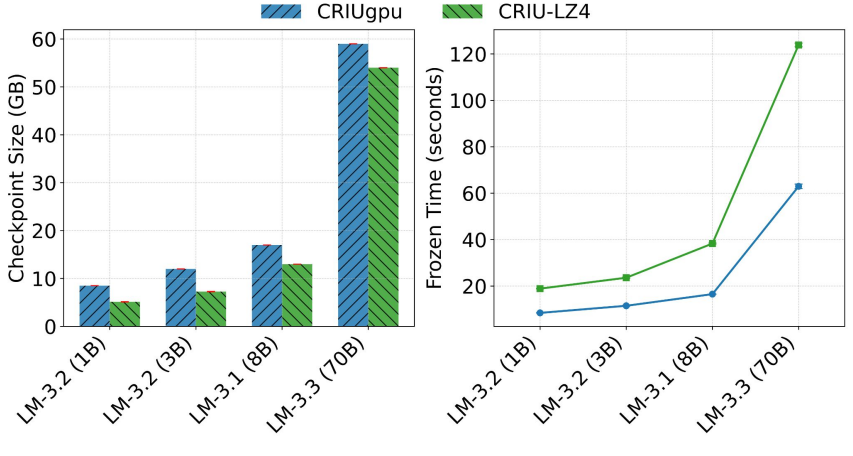
- Memory pages are compressed before being written to storage
- Zero-filled pages are stored as a marker (no data)
- Incompressible pages are stored uncompressed (no decompression overhead)

On-the-Fly Memory Compression



vLLM containers serving Gemma (GM), LLaMA (LM), and DeepSeek-R1 (DS) models on NVIDIA A100 PCIe (40GB)

Container rootfs and checkpoints are stored on in-memory filesystem

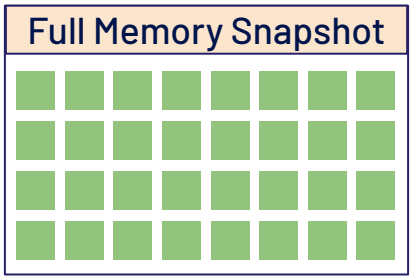


Supervised Fine-Tuning (SFT) workloads for LLaMA models running on NVIDIA B200 SXM6 (180GB)

Optimizing Checkpoints with Built-in Memory Page Compression. Linux Plumbers Conference (2025)

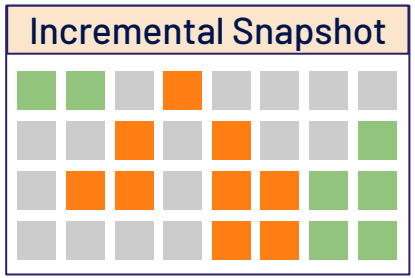
Incremental GPU Checkpointing

Initial Checkpoint



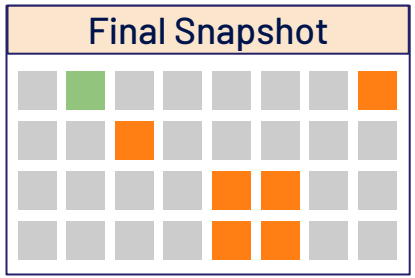
Reset memory tracking via
`/proc/<PID>/clear_refs`

Checkpoint Iteration *N*



- *Scan page table entries*
- *Skip unmodified pages*
- *Reset memory tracking*

Checkpoint Iteration *N+1*



Save only memory changes since last checkpoint

*Key Challenge: CUDA checkpoint **allocates new host pages every time**, all marked as modified, even if content is unchanged.*

Incremental GPU Checkpointing

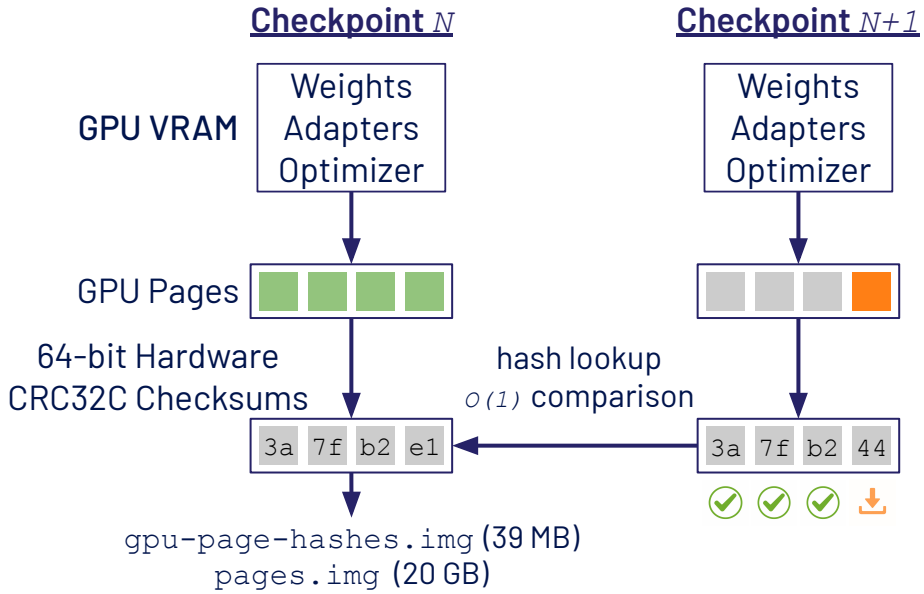
Memory tracking + Content-based Deduplication

Non-GPU Pages
(Python, PyTorch)

Soft-dirty Memory Tracking
Identify modified pages using soft-dirty PTEs

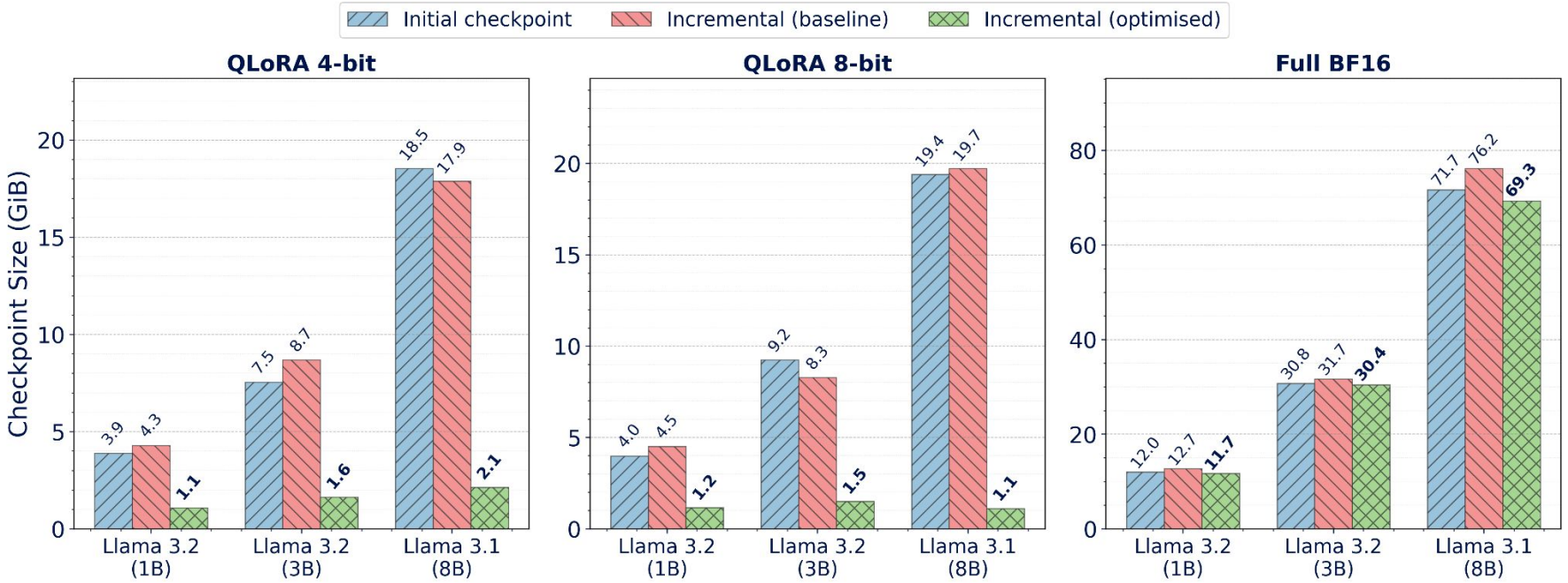
GPU Pages
(driver-managed allocations)

Content Comparison
CRC32C checksum per page, skip matching pages



LaMA-8B QLoRA: 89% pages skipped (20 GB ⇨ 2 GB)

Evaluation Across Fine-Tuning Methods

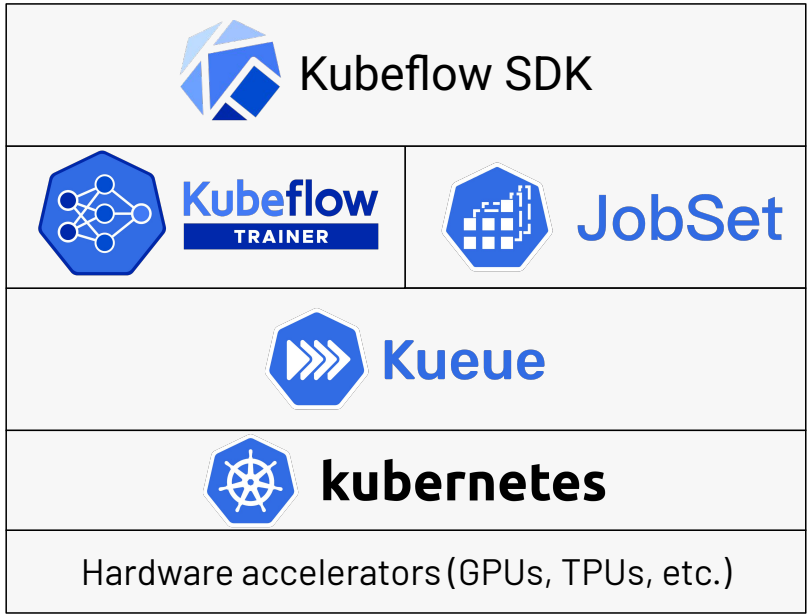


Baseline checkpoints have no optimisations.
Optimised checkpoints use LZ4 compression & content deduplication.
NVIDIA A100 SXM4 (80GB)

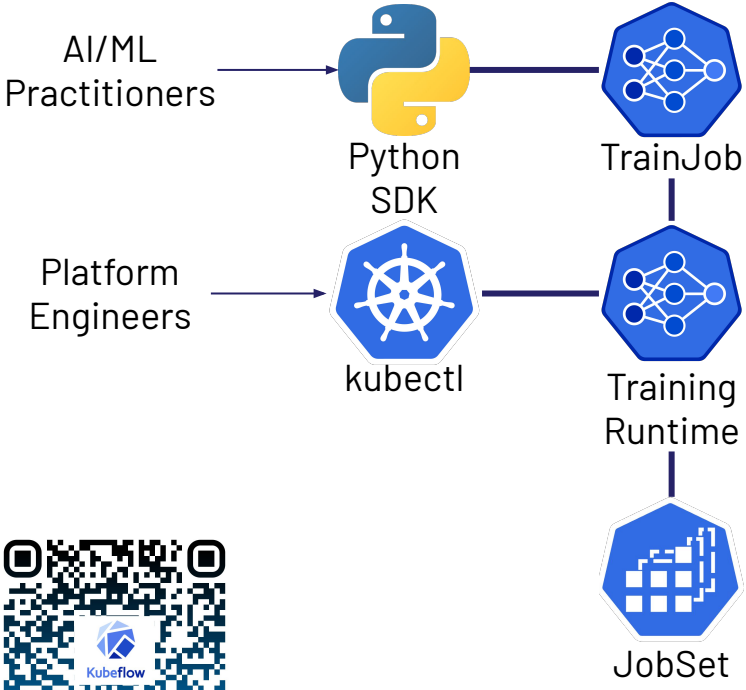
Checkpoint/Restore with Cloud Native AI Systems

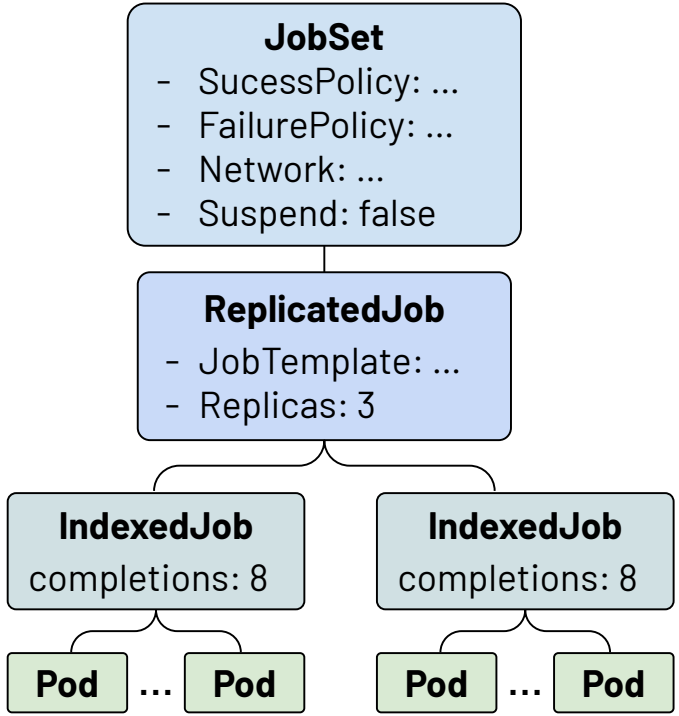
Ecosystem integration with Kubeflow Trainer and Kubernetes

Cloud Native AI Ecosystem

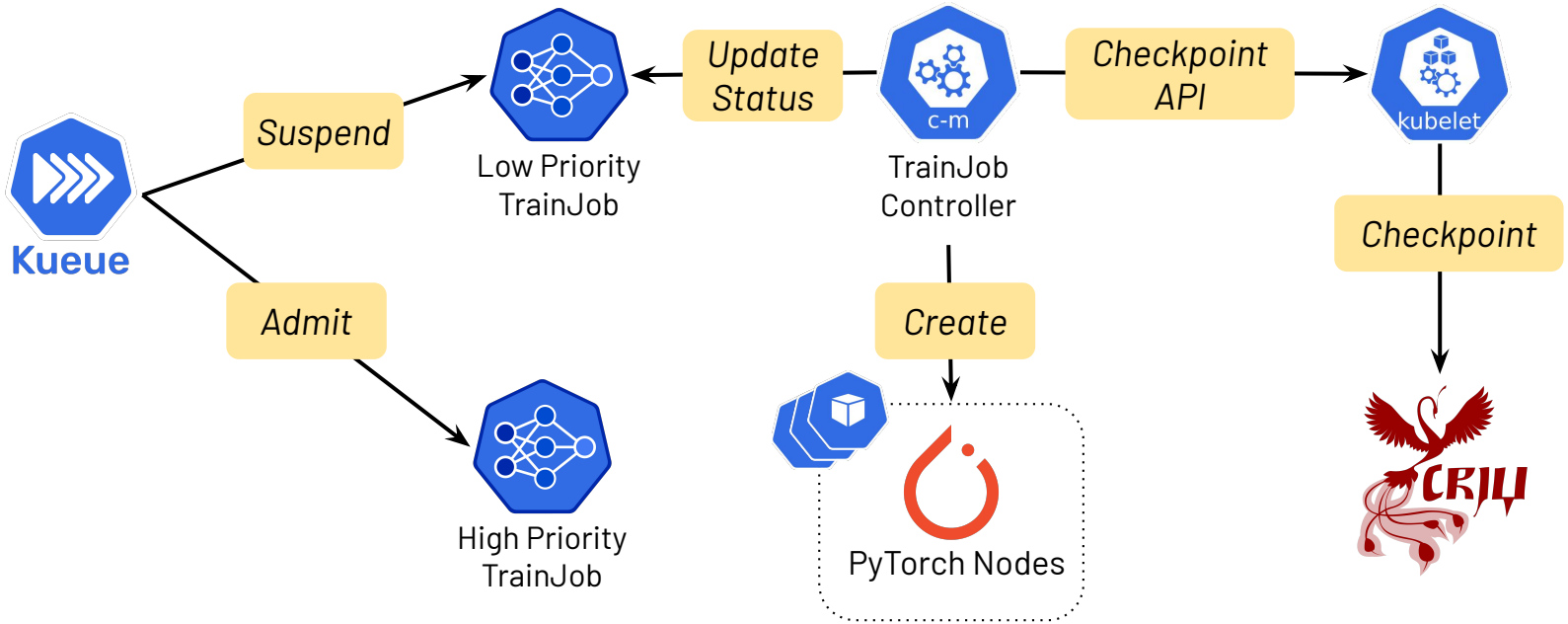


Kubeflow Trainer Overview

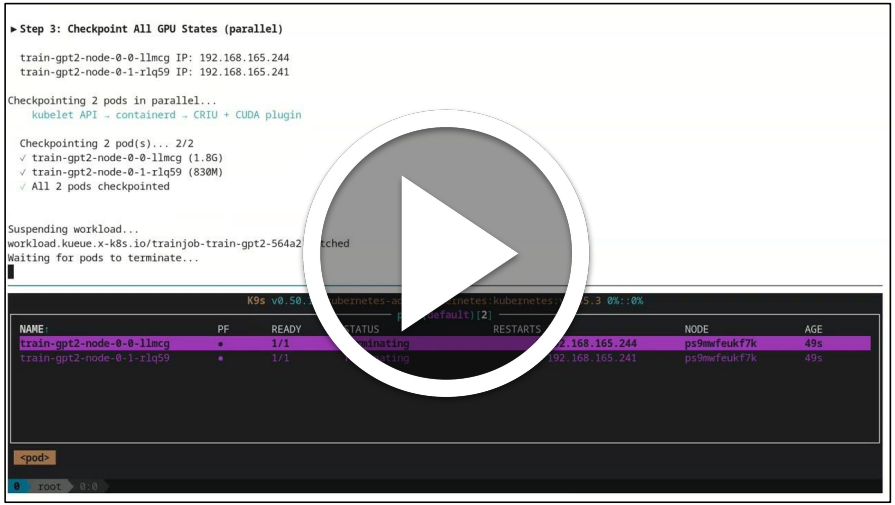




Preemption of Distributed TrainJobs



Checkpoint-Based Suspending of Train Jobs

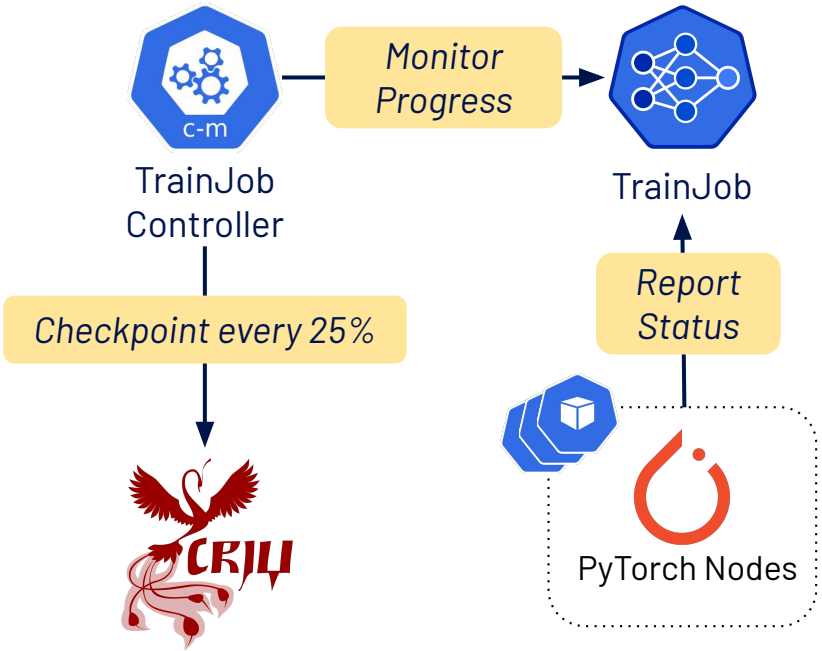


Kueue

Kubeflow TRAINER

CRIU

Future Evolution of TrainJob Checkpointing




```yaml
apiVersion: trainer.kubeflow.org/v1alpha1
kind: TrainJob
...
status:
 trainerStatus:
   lastUpdatedTime: 2025-01-23T10:30:45Z
   # 25% complete
   progressPercentage: 25
   # Precise duration
   estimatedRemainingSeconds: 795649
   # The most recent training metrics that were
reported
   metrics:
     - name: loss
       value: "0.2347"
     - name: accuracy
       value: "0.9876"
```

KEP-2779: Track TrainJob Progress and Expose Training Metrics – Trainer v2.2

Get Involved

Meetings

- Kubernetes Checkpoint Restore WG - Weekly on Thursdays at 5pm (UTC)
- Kubernetes Batch WG - Bi-weekly meetings on Thursdays 2pm (UTC)
- Kubeflow Trainer call - Bi-weekly meetings on Wednesday 6:00am and 9:00am (PST)

Mailing list

- kubernetes.io/g/wg-checkpoint-restore
- kubeflow-discuss
- kubernetes.io/g/wg-batch

Join the CNCF & Kubernetes Slack

- #wg-checkpoint-restore
- #kubeflow-trainer
- #wg-batch


KubeCon
CloudNativeCon
Europe 2026