

TorchPass Snapshots: Platform Snapshots for Distributed Training

October 4, 2026 · clockwork.io/blog/torchpass-snapshots-platform-snapshots-for-distributed-training/

Preserve the state of a running training job without touching the training code.

Large training jobs are interrupted for two kinds of reasons. Some are predictable: a scheduler preempts a job for higher-priority work, a cloud provider reclaims a spot instance, an operator moves a straggling server or rebalances the cluster, or a host is drained for maintenance or because health signals predict a failure. Others are sudden: a GPU, node or kernel fails without warning.

Unfortunately for distributed training jobs, sudden failures are far more common. Meta's Llama 3 pre-training run recorded 466 interruptions in 54 days on 16,384 GPUs. Of these, 419 were unexpected, and about 78 percent of those were attributed to confirmed or suspected hardware issues [1].

The warning before capacity disappears is often short. Amazon Elastic Compute Cloud (EC2) Spot Instances get a best-effort two-minute notice, and Google Cloud Spot virtual machines up to 30 seconds [2–3]. Hardware errors that permit a controlled drain can leave unaffected jobs running until they finish or save state before reset [4]. There is no standard drain deadline: Slurm's DRAIN state allows existing jobs to finish, while Kubernetes provides a configurable grace period that

defaults to 30 seconds once Pod termination begins.

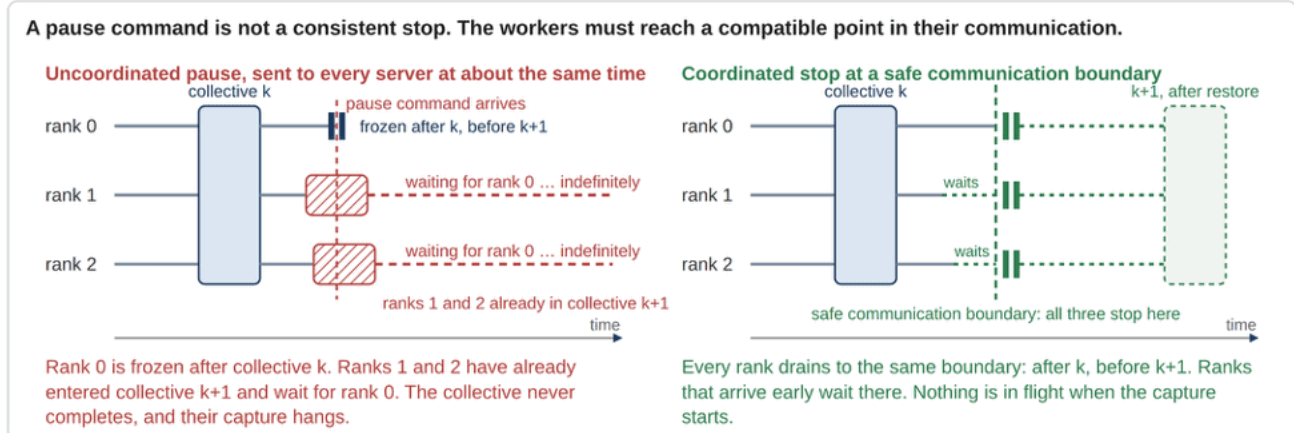
Application checkpoints can automate interruption and restart. But each workload needs a tested integration that saves complete state before termination and reloads it correctly. A cloud operator depends on tenants to provide that response. Platform teams rely on model builder teams to have an integrated, automated response. When that automation is absent or unverified, reclamation means lost work.

TorchPass snapshots let the platform preserve supported distributed training jobs without model-specific checkpoint callbacks. It captures their running process and GPU state, protects the image outside the machines being released, and restores the job when compatible capacity returns.

Replacement GPUs do not have to be ready when the original resource is released.

An industry first, because distributed jobs are hard to snapshot!

NVIDIA's cuda-checkpoint captures supported GPU state at the driver level, and CRIU (Checkpoint/Restore In Userspace) captures the Linux process around it [5]. However, multi-node jobs need additional synchronization across machines (CRIUgpu study [6]). A distributed training job is many workers that wait on one another inside collective operations run by the NVIDIA Collective Communications Library (NCCL). For multi-node snapshots, there are two problems:



1. The first is timing. If one worker freezes just after a collective while its peers have already entered the next one, the peers wait for it indefinitely [7]. For a multi-node snapshot to work, every worker must stop at the same point in its communication, with nothing in flight (Figure 1).
2. The second is the network. Remote direct memory access (RDMA) connections depend on queue pairs and memory keys that the network adapter holds for the running process. Their identifiers can change when those resources are recreated, so a restored copy of the process cannot simply reuse them [8].

To free up machines without losing training progress, or to take a safe snapshot and allow training to continue, the platform needs a coordination layer that pauses all workers at a safe point and saves the snapshot outside the machines being released.

What TorchPass Snapshots do

TorchPass snapshots supply that layer. It uses the same capture-and-restore components as TorchPass live GPU migration, which moves workers from a node (or pod) onto a ready spare. It installs through the job's launch environment, so the training code does not require any changes [9].

Before capture, TorchPass checks that the job is eligible: the GPU, driver and kernel combination, process-capture support, network behavior and required files. Its NCCL interceptor then drains every worker to one safe point in its communication, records the communicator identities and sequence information, and releases the live transport state. Selected outbound HTTP connections are drained and suspended, and eligible files that the destination lacks are staged. A Clockwork optimized cuda-checkpoint and CRIU facility then captures each worker's supported GPU and process state at that exact point.

The snapshot becomes a recovery point. For periodic snapshots, the workers resume in place, while the image is persisted in the background. If a snapshot is later restored, TorchPass brings back process and GPU memory on machines with the same GPU count and a compatible GPU type, driver

and kernel. It then rebuilds the communicators and resumes training from the captured point.

What it costs and where snapshots fit

Internal Clockwork testing on a Megatron-LM test on 8 H200 nodes measured a total pause time of under 20 seconds, with about 72 gibibytes (GiB) of GPU memory per rank. If snapshots are taken every 30 minutes, the pause takes about 1.1 percent of wall-clock time. Every 10 minutes, it takes about 3.3 percent. **For an operator, this means a small performance tax can be imposed in return for a guaranteed safe fallback, without a tenant having to modify their training code.**

Snapshots offer a new layer of protection at the platform level. However, application checkpoints are still valuable. For example, a job moving onto an incompatible driver or kernel needs a checkpoint and a fresh runtime. Checkpoints can also support a different GPU count or parallelism layout where the framework and format allow it.

A snapshot also changes what recovery involves. A model checkpoint restart starts a fresh runtime and reloads the model, and Meta's reliability model assumes 5 to 20 minutes for that initialization [10]. Whereas a restored platform snapshot instead reloads the initialized runtime far faster from its image and rebuilds the communicators.

Snapshots allow a platform team or Neocloud to offer protection of training state without requiring tenants or model builders to change their training code, supporting the following use cases:

- **State capture for preemption and capacity reclamation.** A lower-priority training job can be automatically saved so its GPUs become available for urgent work or can be given to another tenant. The saved job effectively remains "suspended" until compatible capacity is available.
- **State capture for maintenance without an immediate replacement.** A still-running job can be captured before its machines are taken down for repair or reset.

- **Periodic state capture for recovery after an unexpected failure.** Periodic snapshots provide protection when a node or worker fails without warning. A surviving, verified snapshot lets the job resume on compatible capacity.

Sources:

1. Grattafiori, A., et al. (Meta). [“The Llama 3 Herd of Models.”](#) arXiv:2407.21783, 2024.
2. Amazon Web Services. [“Spot Instance interruption notices.”](#) Amazon EC2 documentation.
3. Google Cloud. [“Spot VMs,”](#) section [“Preemption process.”](#) Compute Engine documentation.
4. NVIDIA docs: <https://docs.nvidia.com/deploy/a100-gpu-mem-error-mgmt/error-recovery-and-response-flags.html>
5. NVIDIA. cuda-checkpoint README. GitHub, accessed September 2026.
6. Stoyanov, R., et al. “CRIUgpu: Transparent Checkpointing of GPU-Accelerated Workloads.” arXiv:2502.16631, 2025.
7. Shukla, D., et al. (Microsoft). “Singularity: Planet-Scale, Preemptive and Elastic Scheduling of AI Workloads.” arXiv:2202.07848, 2022, Section 4.3
8. Cao, J., et al. “Transparent Checkpoint-Restart over InfiniBand.” ACM HPDC 2014.
9. Clockwork TorchPass: <https://clockwork.io/wp-content/uploads/2026/03/TorchPass-Whitepaper-From-Taint-Drain-Chkpt-To-Taint-Migrate.pdf>
10. Kokolis, A., et al. (Meta). “Revisiting Reliability in Large-Scale Machine Learning Research Clusters.” IEEE HPCA 2025.