FFUSE: I/O Fault Injection and Resilience Scoring for HPC and AI Workloads

Anila Ghazanfar✉
Julian Kunkel (advisor)
anila.ghazanfar@stud.uni-goettingen.de
julian.kunkel@gwdg.de
Georg August Universität
Göttingen, Lower Saxony, Germany

Abstract

As HPC and AI/ML workloads increasingly share storage, filesystem faults (errors, partial I/O, latency spikes, silent corruption) can cause crashes or undetected data loss. To our knowledge, no systematic comparison of HPC and AI responses to identical filesystem faults, or unified cross-domain metrics, exists. We present FFUSE, a FUSE-based framework injecting seven filesystem fault categories transparently at the application I/O layer, and FIORE, a metric for failures without error reporting. Across 27 runs per workload, LAMMPS (HDF5 checkpoints) had lower crash (29.6% vs 51.9%) and failure (44.4% vs 66.7%) rates than ResNet-50 (pickle-based checkpoints). FIORE was 0.333 (LAMMPS) and 0.222 (ResNet-50), indicating ResNet-50 reported a larger share of its failures. Checkpoint format was associated with differences in I/O-layer fault behavior; a small paired pilot holding the application fixed suggests format can alter recovery outcomes but cannot isolate its effect. These results may inform checkpoint reliability in converged HPC–AI systems.

CCS Concepts

• **Computer systems organization → Reliability**.

Keywords

Fault Injection, FUSE, I/O reliability, Storage reliability, Filesystem, AI workloads, HPC workloads

ACM Reference Format:
Anila Ghazanfar and Julian Kunkel (advisor). 2026. FFUSE: I/O Fault Injection and Resilience Scoring for HPC and AI Workloads. In *Proceedings of November 17–19, 2026 (SC 2026 ACM Student Research Competition).* ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Storage reliability is critical in data centers hosting both High-Performance Computing (HPC) and Artificial Intelligence (AI) workloads, where growing system complexity makes I/O faults such as transient errors, corruption, latency spikes, and partial failures increasingly common. Prior work has characterized how storage faults affect HPC applications [1] and how parallel file systems recover from failures [2]. For AI, existing studies examine process and hardware failures in LLM training [5], hardware failures in deep learning training systems [3], and silent data corruption in distributed training [4], but they do not inject faults at the filesystem interface. Existing tools also typically test isolated fault types, whereas real storage degradation often involves cascading failures, and no prior framework compares HPC and AI workloads under identical filesystem-level faults.

We present FFUSE, a FUSE-based fault injection framework that transparently injects seven categories of faults at the POSIX I/O layer without modifying applications. We evaluate LAMMPS (HPC, HDF5 checkpoints) and PyTorch ResNet-50 (AI, pickle-based checkpoints) under identical conditions, with 26 fault-injection tests per workload. We ask (RQ1) whether the two domains differ in tolerance to filesystem faults, (RQ2) which fault types pose the greatest risk to each, and (RQ3) what domain-specific hardening this requires. To quantify fault awareness at the I/O layer, we introduce FIORE (Fault Injection Observed Resilience Evaluation), which measures the proportion of failures that go unreported (silent failures), and we report it alongside reporting coverage (complement of FIORE), crash rate, failure rate, silent failure rate, error detection rate, and data integrity rate.

2 Methodology

The FFUSE Framework. FFUSE uses FUSE (Filesystem in Userspace) to intercept POSIX I/O calls transparently, allowing fault injection without modifying application source code. The architecture (Figure 1) comprises three logical layers: the application layer, where standard workloads such as LAMMPS and ResNet-50 issue checkpoint and data-access calls; the kernel layer, where VFS and the FUSE kernel module forward these calls to userspace; and the userspace-storage layer, where a custom daemon injects faults such as delays, corruption, and error codes before forwarding requests to the underlying filesystem.

Seven-Phase Fault Taxonomy. Unlike prior tools [1] that evaluate isolated faults in isolation, FFUSE models realistic storage degradation through seven phases (Figure 2): baseline operation, fatal error injection (e.g., EIO, ENOMEM, ENOSPC), short I/O, transient errors, cascading faults, delayed injection, and silent corruption. Together, these phases capture both explicit failures and silent integrity loss that arise during checkpointing and model persistence.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SC 2026 ACM Student Research Competition,
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

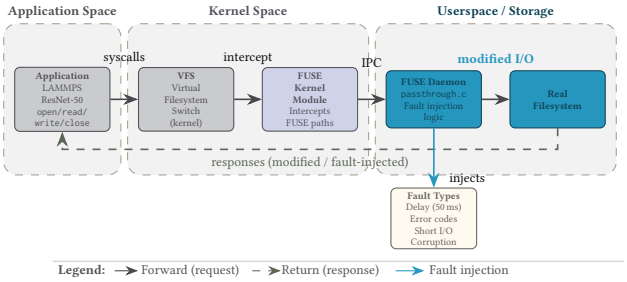


Figure 1: FUSE architecture with fault injection.

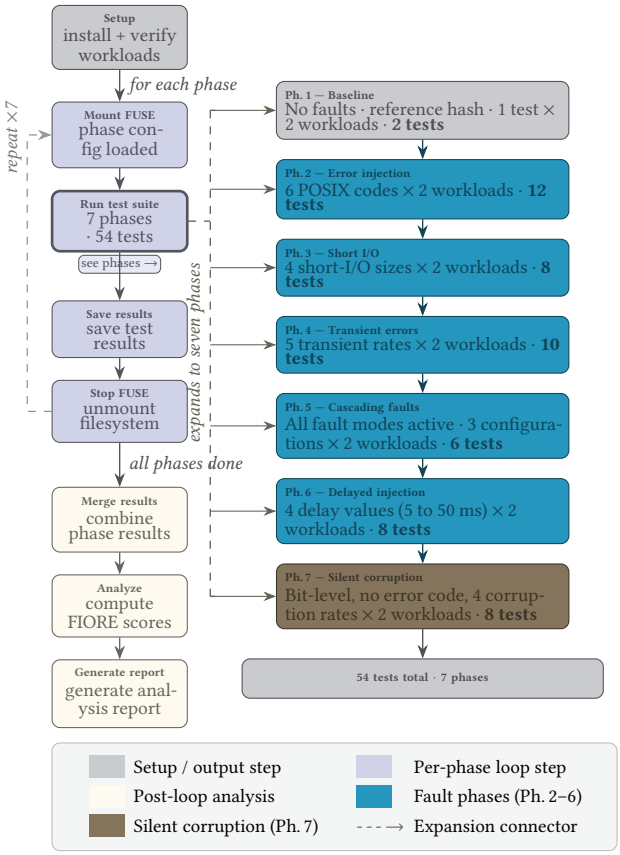


Figure 2: FFUSE execution workflow.

3 Evaluation Metrics

We report five outcome metrics per workload and fault phase (Table 1), plus FIORE, which measures how many failures go unreported. Rates are percentages of N_{total} runs. A run counts as a failure if it crashes or has a confirmed silent failure, counted once, and it is reported if an application error report is recorded.

FIORE.. FIORE quantifies how often failures occur *without* an application error report:

$$\text{FIORE} = \frac{N_{\text{failure}} - N_{\text{reported}}}{N_{\text{failure}}}. \tag{1}$$

Table 1: Evaluation metrics (N_{total} = runs; N_{silent} excludes crashed runs).

Metric	Definition
Crash rate	$N_{\text{crash}}/N_{\text{total}}$: runs that terminated abnormally
Failure rate	$(N_{\text{crash}} + N_{\text{silent}})/N_{\text{total}}$: reported and silent failures
Silent failure rate	$N_{\text{silent}}/N_{\text{total}}$: fault injected, no crash, no reported error
Error detection rate	$N_{\text{reported}}/N_{\text{total}}$: runs where the application surfaced an error
Data integrity rate	$(N_{\text{hash_ok}} + N_{\text{op_ok}})/N_{\text{total}}$: output matches the fault-free baseline (SHA-256 for complete artifacts, integrity flag for incomplete ones)

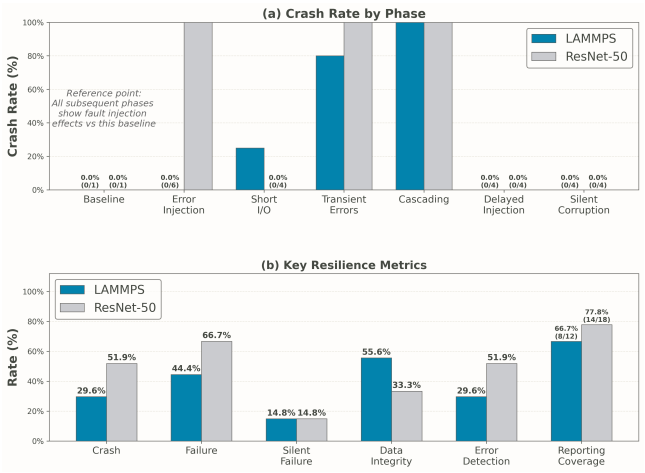


Figure 3: LAMMPS and ResNet-50 summary: (a) crash rate by phase; (b) key metrics.

A higher FIORE means a larger share of failures went unreported. Reporting coverage is $N_{\text{reported}}/N_{\text{failure}} = 1 - \text{FIORE}$. If no failures occur, both are N/A and the failure rate is 0%.

4 Experimental Setup

We evaluated two representative workloads under the same FFUSE fault injector and runtime conditions: LAMMPS, a molecular dynamics workload that writes HDF5 checkpoints, and ResNet-50, a deep-learning training workload that writes PyTorch `.pt` checkpoints. Each workload was evaluated across seven phases for 27 runs, yielding 54 total experiments. This unified setup enables direct comparison of cross-domain resilience under matched I/O-layer faults.

5 Results

LAMMPS had a lower crash rate than ResNet-50 (8/27, 29.6% vs. 14/27, 51.9%) and a lower failure rate (12/27, 44.4% vs. 18/27, 66.7%) (Figure 3). FIORE was 0.333 for LAMMPS (4 of 12 failures unreported; 95% Wilson CI [0.138, 0.609]) and 0.222 for ResNet-50 (4 of 18; [0.090, 0.452]). The intervals overlap, so these data cannot separate the workloads on FIORE.

Shared vulnerabilities. Silent corruption was the most critical: both workloads showed a 0% crash rate, yet corruption was caught only by post-execution hash comparison, never by an application error report. Transient faults crashed both workloads (LAMMPS 4/5, ResNet-50 5/5), and both failed under all cascading faults (3/3 each).

Differences and checkpoint format. LAMMPS (HDF5) tolerated short I/O and latency faults, whereas ResNet-50 (pickle-based `.pt`) failed under short I/O and transient write faults. With two workloads, this is an association between format and behavior; it cannot separate format from application.

Format pilot. To probe this, we fixed one ResNet-50 CIFAR-10 training/resume path and varied the serializer (`.pt`, HDF5, SafeTensors), with one trial per format and fault scenario (15 cases). All formats recovered exactly at baseline and none did under injected faults. Under silent corruption, `.pt` and SafeTensors resumed with a diverged final model, while HDF5 failed to reopen; short I/O, transient errors, and EIO failed all formats. Format can therefore change recovery outcomes, but the pilot is small (n=1 per cell) and cannot show that format is the primary determinant of I/O-layer resilience; that needs exposure-matched, multi-seed, full-duration runs.

6 Conclusion

Across two workloads, silent corruption was the shared, unreported I/O-layer failure mode, and checkpoint format was associated with differences in recovery behavior. HPC–AI systems may benefit from validation-aware checkpointing and end-to-end integrity checks. Future work includes the exposure-matched format ablation, distributed storage (Lustre/GPFS) and GPU-memory faults (current results are CPU-only), and CI/CD integration for resilience regression testing. FFUSE is open source: https://github.com/anila-gitUser/FFUSE/.

Acknowledgment

The authors would like to thank the German Federal Ministry of Research, Technology and Space and the German federal states (http://www.nhr-verein.de/en/our-partners) for supporting this work as part of the National High-Performance Computing (NHR) joint funding program.

References

[1] B. Fang, D. Wang, S. Jin, Q. Koziol, Z. Zhang, Q. Guan, S. Byna, S. Krishnamoorthy, and D. Tao. 2021. Characterizing Impacts of Storage Faults on HPC Applications: A Methodology and Insights. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 409–420. doi:10.1109/CLUSTER48925.2021.00048

[2] R. Han, O. Gatla, M. Zheng, J. Cao, D. Zhang, D. Dai, Y. Chen, and J. Cook. 2022. A Study of Failure Recovery and Logging of High-Performance Parallel File Systems. *ACM Transactions on Storage* 18 (2022), 1–44. doi:10.1145/3483447

[3] Y. He, M. Hutton, S. Chan, R. De Gruijl, R. Govindaraju, N. Patil, and Y. Li. 2023. Understanding and Mitigating Hardware Failures in Deep Learning Training Systems. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA '23)*. ACM. doi:10.1145/3579371.3589105

[4] E. Rojas, D. Pérez, and E. Meneses. 2022. Exploring the Effects of Silent Data Corruption in Distributed Deep Learning Training. In *2022 IEEE 34th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 21–30. doi:10.1109/SBAC-PAD55451.2022.00013

[5] B. Wu, L. Xia, Q. Li, K. Li, X. Chen, Y. Guo, T. Xiang, Y. Chen, and S. Li. 2023. *TRANSOM: An Efficient Fault-Tolerant System for Training LLMs.* arXiv:2310.10046 doi:10.48550/arXiv.2310.10046

Received 8 August 2026; revised 30 September 2026; accepted 0 September 2026