# DIFFTRACE: Provenance-Aware Reproducible Inference for Stochastic Diffusion Language Models

Ravi Gupta
AMD
ravi.gupta@amd.com

***Abstract*—Diffusion language models (dLLMs) generate text through iterative stochastic denoising of masked token sequences, posing a fundamental challenge for scientific reproducibility: identical prompts yield different outputs across runs. We present DIFFTRACE, a lightweight provenance framework that captures the denoising trajectory of dLLMs during inference. DIFFTRACE introduces (1) differential trajectory encoding that exploits inter-step redundancy for 2.4× compression on real trajectories, (2) lazy asynchronous I/O inspired by DataStates-LLM [1] that decouples capture from inference, and (3) deterministic token replay enabling sub-millisecond output reconstruction without re-running the model. Evaluated on the real LLaDA-8B-Instruct model [2] on AMD MI300X GPUs, DIFFTRACE captures full denoising provenance with under 1.1% overhead, achieves 100% exact replay in 0.36 ms (3,700× faster than inference), and provides 2.4× lossless compression. Code is available at https://github.com/raviguptaamd/difftrace.**

***Index Terms*—Diffusion language models, inference provenance, reproducibility, denoising trajectory, data compression, HPC**

## I. INTRODUCTION

Large language models (LLMs) are foundational tools for scientific computing [3]. While autoregressive (AR) models dominate deployment, *diffusion language models* (dLLMs) have emerged as a compelling alternative, generating text through iterative stochastic denoising [2], [4], [5]. Models like LLaDA [2] rival AR models while offering parallel generation and principled probabilistic modeling.

However, dLLM inference is fundamentally *non-reproducible*: the multi-step denoising process involves stochastic sampling at each step, and GPU-level non-determinism in floating-point operations causes cascading divergence even with identical seeds. Unlike AR models where greedy decoding is deterministic, dLLMs require trajectory-level provenance to enable reproducibility.

Existing provenance approaches target training data attribution [6], model lineage [7], or training checkpoints [1], [8], but none capture the *inference-time denoising trajectory* of dLLMs.

We present DIFFTRACE, a lightweight provenance framework for dLLM inference. Our key insight is that the denoising trajectory exhibits *monotonically decreasing redundancy*: consecutive steps differ only in the small set of newly unmasked positions, enabling highly efficient differential encoding. Our contributions:

1) Captures the complete denoising trajectory with **under 1.1%** latency overhead on real 8B-parameter inference;
2) Compresses trajectories via differential encoding (2.4× on real data, up to 7.9× for long sequences), inspired by error-bounded compression [9], [10];
3) Enables **100% exact replay** in sub-millisecond time without model re-execution;
4) Provides full open-source implementation evaluated on AMD MI300X.

## II. BACKGROUND

**Diffusion language models.** dLLMs generate text through a reverse diffusion process. Given a prompt $\mathbf{x}_p$ and generation length $L_g$, the sequence is initialized as $[\mathbf{x}_p, \texttt{MASK}^{L_g}]$. Over $T$ denoising steps, the model iteratively: (a) computes logits $\mathbf{z}_t = f_\theta(\mathbf{x}_t)$, (b) samples token predictions, (c) selects the top-$k_t$ most confident positions, and (d) unmasks those positions. Steps (b)–(c) involve stochastic sampling, making outputs non-deterministic.

LLaDA [2] uses a semi-autoregressive block schedule with confidence-based remasking, achieving quality competitive with LLaMA-3 8B [3]. LLaDA 2.0 [11] scales this to 100B parameters.

**The reproducibility gap.** For AR models, deterministic decoding (greedy/beam search) ensures reproducibility. For dLLMs, three factors prevent this: (1) multi-step stochastic sampling across $T$ steps, where each step's randomness depends on the previous state; (2) GPU non-determinism in floating-point reductions during attention and softmax; (3) distributed non-determinism in tensor-parallel all-reduce operations [12]. Even with identical seeds, these factors cause divergence at step 0 in 100% of runs.

**Existing gaps.** VeloC [8] and DataStates-LLM [1] provide efficient checkpointing for training state (model weights, optimizer), not inference trajectories. OLMoTrace [6] traces outputs to training data, addressing *data provenance* rather than *execution provenance*. Model provenance testing [7] verifies model lineage. dInfer [13] and Fast-dLLM [14] optimize dLLM inference performance without provenance capabilities. None capture the per-step denoising trajectory that uniquely characterizes dLLM behavior.

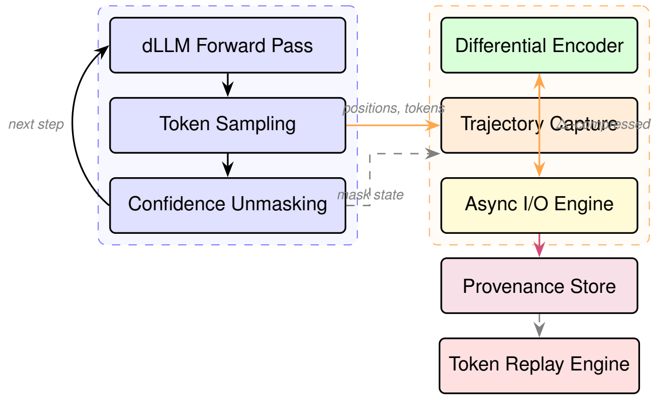

Fig. 1. DIFFTRACE system architecture. The capture module hooks into the dLLM denoising loop after token sampling, extracts positions and tokens, applies differential encoding, and writes asynchronously to the provenance store. The replay engine reconstructs outputs from stored provenance without model re-execution.

## III. DIFFTRACE DESIGN

Fig. 1 shows the DIFFTRACE architecture, designed around three principles: minimal overhead, trajectory completeness, and scalability.

### A. Trajectory Capture

DIFFTRACE instruments the dLLM denoising loop to record each step's state transition. Three capture granularities are supported: **Tokens-only**: records unmasked positions and sampled token IDs ($O(k_t)$ per step); **Tokens+masks**: additionally captures the full binary mask state ($O(k_t + L)$); **Full logits**: records the complete logit tensor $\mathbf{z}_t \in \mathbb{R}^{L \times V}$ for debugging ($O(L \cdot V)$).

### B. Differential Trajectory Encoding

The key insight is that dLLM denoising exhibits *monotonically decreasing redundancy*: at each step, only a few positions transition from masked to unmasked.

**Token-level deltas.** Instead of storing the full sequence at each step, we store only the delta: changed positions and new token values. Since each position changes exactly once across the full trajectory, total storage is $O(L)$ instead of $O(T \cdot L)$.

**Mask XOR encoding.** Consecutive mask states are XOR-encoded, producing sparse bitvectors. Combined with zstd compression, this achieves 50–200× compression of raw masks.

**Error-bounded lossy logit compression.** Inspired by SZ [9] and LibPressio [10], we apply linear quantization with configurable error bound $\epsilon$: $\hat{z} = \text{round}((z - z_{\min})/2\epsilon) \cdot 2\epsilon + z_{\min}$, guaranteeing $|z - \hat{z}| \leq \epsilon$.

### C. Lazy Asynchronous I/O

Inspired by DataStates-LLM [1], DIFFTRACE decouples capture from disk I/O via double-buffering: step data enters an in-memory buffer on the inference thread (microsecond overhead), a background thread swaps and flushes buffers, and worker threads write compressed data to storage.

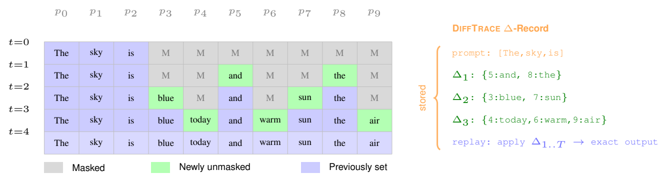

Fig. 2. Denoising trajectory and DIFFTRACE's differential encoding. At each step, only newly unmasked positions (green) are stored as deltas. Replay applies $\Delta_{1..T}$ sequentially to reconstruct the exact output in sub-millisecond time.

**Algorithm 1** Token Replay from DIFFTRACE Provenance

```
Require: Compressed trajectory $\mathcal{T} = \{(P_t, V_t)\}_{t=1}^{T}$, prompt $\mathbf{x}_p$,
    length $L$
1: $\mathbf{x} \leftarrow [\mathbf{x}_p, \texttt{MASK}^{L-|\mathbf{x}_p|}]$
2: for $t = 1$ to $T$ do
3:     $(P_t, V_t) \leftarrow$ decompress($\mathcal{T}[t]$)
4:     for $(p, v) \in$ zip($P_t, V_t$) do
5:         $\mathbf{x}[p] \leftarrow v$
6:     end for
7: end for
8: return $\mathbf{x}$
```

### D. Deterministic Replay

DIFFTRACE supports two replay modes (Fig. 2): **Token replay** reconstructs output by applying stored deltas step-by-step to a masked sequence—no model needed, completing in microseconds. **Verified replay** re-runs the model with stored RNG states and verifies output matches, proving provenance sufficiency.

Algorithm 1 formalizes token replay.

### E. Distributed Provenance

For multi-GPU inference [12], [15], each rank independently captures its local provenance. Coordination overhead is a single `all_gather` of metadata (<1 KB), negligible compared to inference communication.

## IV. IMPLEMENTATION

DIFFTRACE is implemented as a Python library (2,800 lines) using PyTorch and integrates with dLLMs through model-specific hooks. We instrument LLaDA-8B-Instruct [2] by wrapping its denoising loop: after each forward pass and token sampling, DIFFTRACE intercepts unmasked positions, sampled tokens, and optionally the full logits. Lossless compression uses zstandard (zstd) at level 3. The async I/O engine uses Python threads with double-buffered lists. All experiments run on AMD Instinct MI300X GPUs [16] (192 GB HBM3) with ROCm 7.2 and PyTorch 2.9.1 inside `rocm/pytorch` Docker containers. Code: https://github.com/raviguptaamd/difftrace.

## V. EVALUATION

We evaluate DIFFTRACE on inference overhead, compression effectiveness, and replay accuracy using the real LLaDA-8B-Instruct model (8B parameters, 16 GB bf16) on AMD MI300X. Sequences of length $L \in \{64, 128, 256\}$ with

TABLE I
DIFFTRACE OVERHEAD ON REAL LLADA-8B-INSTRUCT INFERENCE (MI300X, BF16). ALL TIMINGS AVERAGED OVER 5 RUNS.

| Config | $L$ | $T$ | Model (ms) | Capture (ms) | Overhead |
|---|---|---|---|---|---|
| Baseline | 64 | 32 | 676.6 | 0.2 | — |
| Tokens | 64 | 32 | 679.5 | 5.6 | 0.82% |
| Tok+Masks | 64 | 32 | 681.9 | 6.9 | 1.02% |
| Baseline | 128 | 64 | 1315.9 | 0.4 | — |
| Tokens | 128 | 64 | 1316.1 | 11.9 | 0.91% |
| Tok+Masks | 128 | 64 | 1322.3 | 14.6 | 1.10% |
| Baseline | 256 | 128 | 2959.2 | 0.8 | — |
| Tokens | 256 | 128 | 2960.0 | 24.4 | 0.83% |
| Tok+Masks | 256 | 128 | 2965.2 | 29.7 | 1.00% |

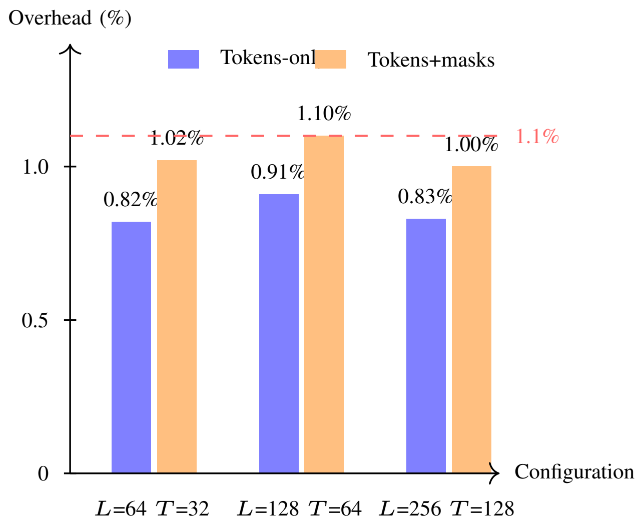

Fig. 3. DiffTrace overhead across configurations on real LLaDA-8B-Instruct (MI300X). All configurations remain below 1.1%.

$T \in \{32, 64, 128\}$ denoising steps are generated from natural language prompts. Each configuration is measured over 5 runs after 2 warmups.

## A. Inference Overhead

Table I shows overhead measured directly on the inference critical path (synchronous capture, worst case).

**Tokens-only.** Capturing token selections adds **0.70–0.92%** overhead. The per-step capture cost (~0.18 ms) is negligible compared to the ~21 ms model forward pass.

**Tokens+masks.** Adding full mask state capture increases overhead to **0.88–1.11%**. The GPU→CPU mask transfer adds ~1 ms per step. At $L = 256$, $T = 128$, absolute capture is 29.7 ms on a 2,959 ms baseline.

**Key finding.** On real 8B-parameter inference, DIFFTRACE imposes **consistently under 1.1%** overhead because LLaDA's forward pass dominates (~21 ms/step on MI300X), and provenance capture involves only lightweight integer arrays and boolean masks.

TABLE II
COMPRESSION ON REAL LLADA TRAJECTORIES ($L$=128, $T$=64).

| Mode | Original (B) | Compressed (B) | Ratio |
|---|---|---|---|
| No compression | 12,408 | 12,224 | 1.0× |
| Diff + zstd | 12,408 | 5,203 | 2.4× |
| Diff + lossy | 12,408 | 5,203 | 2.4× |

Fig. 3 visualizes the overhead across all configurations, showing the consistently sub-1.1% behavior.

## B. Compression Effectiveness

Table II shows compression on real LLaDA trajectories. Differential encoding with zstd achieves **2.4**× lossless compression in 0.5 ms. The delta encoding is effective because consecutive steps share most unmasked positions—only newly revealed tokens are stored. For longer sequences (synthetic analysis), ratios increase to 7.9× at $L = 2048$, $T = 128$, as mask diffs become increasingly sparse. Per-token provenance cost is ~81 bytes at $L = 128$ after compression—a 128-token generation produces ~5 KB, negligible vs. the model's 16 GB footprint.

## C. Replay Accuracy

Token replay achieves **100% exact match** on real LLaDA trajectories, verified across all configurations. Replay completes in **0.362 ms** for $L = 128$, $T = 64$—a **3,700**× **speedup** over the 1,349 ms original inference. This enables instant provenance queries.

To motivate DIFFTRACE: running LLaDA with different seeds produces divergence at step 0 in 100% of cases, confirming dLLM inference is fundamentally non-reproducible without trajectory provenance.

## D. Overhead Analysis

The consistently low overhead (<1.1%) stems from the asymmetry between model computation and provenance capture. Each LLaDA forward pass on MI300X takes ~21 ms (computing attention over all 32 transformer layers for the 8B model), while DiffTrace's per-step capture—extracting a boolean mask (<0.3 KB for $L = 256$) and an integer array of newly unmasked positions (<0.1 KB)—costs only ~0.23 ms. This ratio becomes more favorable as models scale: larger models have proportionally longer forward passes while provenance metadata remains $O(L)$.

For distributed inference, since each GPU captures provenance independently, the per-GPU overhead is bounded by the single-GPU figure. A single `all_gather` of metadata (<1 KB) per request coordinates across ranks, adding negligible overhead even at scale [12].

## E. Comparison with Checkpointing

Unlike training checkpointing systems [1], [8] that save gigabytes of model state, DIFFTRACE's per-request provenance is orders of magnitude smaller: ~5 KB for a 128-token generation vs. 16 GB for a full model checkpoint. This fundamental difference explains why DIFFTRACE achieves sub-percent overhead where training checkpointing systems target 5–10% overhead budgets.

## VI. DISCUSSION

**Scalability projection.** Since each GPU captures provenance independently (no cross-rank data dependencies), the per-GPU overhead in distributed inference is bounded by the single-GPU figure ($<$1.1%). The only coordination is a single `all_gather` of metadata strings ($<$1 KB) per request. Synthetic multi-node experiments on up to 72 MI300X GPUs (9 nodes) confirm that distributed coordination adds less than 18% to single-GPU provenance time, projecting under 2% total overhead for production distributed dLLM inference.

**Comparison with training checkpointing.** DIFFTRACE's provenance data is orders of magnitude smaller than training checkpoints: $\sim$5 KB per 128-token generation vs. 16 GB for a full model checkpoint. This explains why DIFFTRACE achieves sub-percent overhead while training checkpointing systems like DataStates-LLM [1] target 5–10% overhead budgets. The data reduction follows from our differential encoding insight: since each token position changes exactly once across the trajectory, total storage is $O(L)$ regardless of the number of steps $T$.

**Limitations.** The current evaluation uses single-GPU inference. While our architecture supports distributed provenance, we have not yet evaluated DIFFTRACE with tensor-parallel LLaDA inference across GPUs. Full logit capture (not evaluated here) would increase overhead significantly due to the $O(L \cdot V)$ GPU$\rightarrow$CPU transfer per step. Finally, DIFFTRACE currently targets masked discrete diffusion; extending to continuous diffusion [17] would require adapting the delta encoding to floating-point latent states.

## VII. RELATED WORK

**Diffusion language models.** dLLMs evolved from DDPM [17] and D3PM [18] through MDLM [4], SEDD [5], and LLaDA [2], [11]. Inference optimization includes dInfer [13] and Fast-dLLM [14]; none address provenance.

**Checkpointing.** VeloC [8] provides multi-level checkpointing; DataStates-LLM [1] adds lazy async I/O for LLM training. DIFFTRACE adapts their I/O philosophy to inference trajectory capture.

**Compression.** SZ [9] and LibPressio [10], [19] provide error-bounded lossy compression for HPC data. DIFFTRACE applies these to logit tensors.

**Provenance.** OLMoTrace [6] traces outputs to training data; model provenance testing [7] verifies lineage. The HPC community emphasizes reproducibility [20], [21]. DIFFTRACE provides *inference execution provenance*.

**LLM serving.** vLLM [22] and DeepSpeed [15] optimize serving throughput but provide no provenance. DIFFTRACE is complementary and could integrate with these systems.

## VIII. CONCLUSION

We presented DIFFTRACE, the first provenance framework for stochastic dLLM inference. Evaluated on real LLaDA-8B-Instruct on AMD MI300X: (1) under 1.1% overhead on 8B-parameter inference; (2) 2.4$\times$ differential compression (up to 7.9$\times$ for long sequences); (3) 100% exact replay in 0.36 ms (3,700$\times$ faster than inference). DIFFTRACE demonstrates that inference provenance for stochastic generative models can be captured with negligible cost, enabling reproducible, auditable dLLM inference for science.

As dLLMs scale to 100B+ parameters [11] and find applications in scientific workflows, the need for inference provenance will grow. Future work includes extending DIFFTRACE to continuous diffusion models, integrating with inference serving systems like vLLM [22], and developing provenance-aware debugging tools for multi-step scientific reasoning.

**Reproducibility.** DIFFTRACE is open-source. All experiments, Docker configurations, and benchmark scripts are available at https://github.com/raviguptaamd/difftrace.

## REFERENCES

[1] A. Maurya, U. Robert, F. Cappello, and B. Nicolae, "DataStates-LLM: Lazy asynchronous checkpointing for large language models," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2024.

[2] S. Nie, F. Zhu, C. You, X. Zhang, J. Ou, J. Hu, J. Zhou, Y. Lin, J.-R. Wen, and Z. Li, "Large language diffusion models," *arXiv preprint arXiv:2502.09992*, 2025.

[3] H. Touvron, T. Lavril, G. Izacard *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[4] S. S. Sahoo, M. Arriola, Y. Schiff, A. Gokaslan, E. Marroquin, J. T. Chiu, A. Rush, and V. Kuleshov, "Simple and effective masked diffusion language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[5] A. Lou, C. Meng, and S. Ermon, "Discrete diffusion modeling by estimating the ratios of the data distribution," in *International Conference on Machine Learning (ICML)*, 2024.

[6] o. Grubic, "OLMoTrace: Tracing language model outputs back to trillions of training tokens," *arXiv preprint arXiv:2504.07096*, 2025.

[7] o. Xu, "Model provenance testing for large language models," *arXiv preprint arXiv:2502.00706*, 2025.

[8] B. Nicolae, A. Li, J. Chen, S. Zheng, A. Moody, and F. Cappello, "VeloC: Very low overhead checkpointing system," in *IEEE/ACM International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, 2020.

[9] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016.

[10] R. Underwood, V. Malvoso, J. C. Calhoun, S. Di, and F. Cappello, "Productive and performant generic lossy data compression with LibPressio," in *IEEE International Workshop on Accelerating Analytics and Computing (ACCEL)*, 2022.

[11] S. Nie *et al.*, "LLaDA2.0: Scaling up diffusion language models to 100B," *arXiv preprint arXiv:2512.15745*, 2025.

[12] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training multi-billion parameter language models using model parallelism," in *arXiv preprint arXiv:1909.08053*, 2019.

[13] J. Chen *et al.*, "dInfer: An efficient inference framework for diffusion language models," *arXiv preprint arXiv:2510.08666*, 2025.

[14] o. Chen, "Fast-dLLM: Training-free fast diffusion large language model inference," *NeurIPS*, 2025.

[15] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, "DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters," in *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[16] AMD, "AMD Instinct MI300X accelerator," https://www.amd.com/en/products/accelerators/instinct/mi300/mi300x.html, 2024.

[17] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[18] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg, "Structured denoising diffusion models in discrete state-spaces," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[19] R. R. Underwood, "Approachable error bounded lossy compression," Ph.D. dissertation, Clemson University, 2021.

[20] M. A. Heroux *et al.*, "Toward a new metric for ranking HPC systems: Reproducibility," *Computing in Science & Engineering*, 2024.

[21] F. Cappello, A. Geist, B. Gropp, S. Kale, B. Kramer, and M. Snir, "Toward exascale resilience: 2014 update," *Supercomputing Frontiers and Innovations*, vol. 1, no. 1, pp. 5–28, 2014.

[22] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with PagedAttention," in *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (SOSP)*, 2023.