ND-Locate: Localizing Non-Determinism in MPI Applications

Dhroov Pandey[1], Jack Marquez[2], Michela Taufer[2], Sanjukta Bhowmick[1]
[1]University of North Texas, Denton, TX, USA [2]University of Tennessee, Knoxville, TN, USA

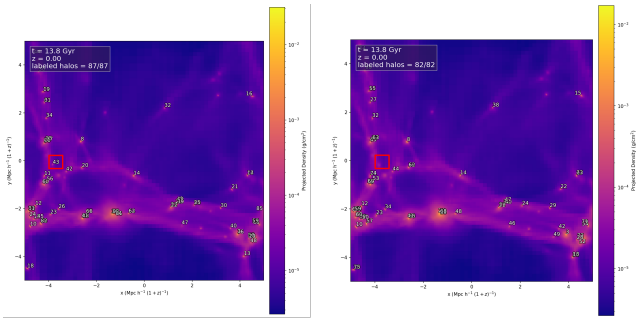
Abstract

Large-scale MPI applications can yield different results from identical inputs due to non-deterministic execution, yet current tools offer little help in pinpointing its source. We present ND-Locate, a framework that localizes execution-level non-determinism to application functions, achieving Pearson correlations up to 0.97 on benchmark applications and identifying the code regions responsible for output variability in the cosmological simulator *Enzo*.

ACM Reference Format:
Dhroov Pandey[1], Jack Marquez[2], Michela Taufer[2], Sanjukta Bhowmick[1]. 2026. ND-Locate: Localizing Non-Determinism in MPI Applications. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '26).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction and Motivation

Scientific simulations should be reproducible. Yet large-scale simulations on distributed-memory systems can yield different results from identical inputs because asynchronous communication introduces non-deterministic execution. Since floating-point arithmetic is not associative, different message arrival orders change the sequence of operations, producing different numerical results and potentially different scientific conclusions. Figure 1 shows this in the Enzo cosmological simulator [3]: two runs with identical initial conditions produce different halo counts (82 vs. 87).



(a) Execution 1 (82 halos) **(b) Execution 2 (87 halos)**

Figure 1: Two Enzo executions with identical initial conditions produce different halo counts (82 vs. 87), demonstrating that non-deterministic execution can alter outcomes.

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 '26, Chicago, IL
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

While detecting that two executions differ is important, the core challenge is pinpointing *where* non-determinism arises in the application. Existing tracing and analysis tools can reveal execution differences but offer little insight into which functions are responsible [2]. As a result, developers must manually inspect millions of events, complicating debugging, validation, and reproducibility.

To address this challenge, we present ND-Locate, a framework that attributes execution-level non-determinism to application functions. ND-Locate traces MPI executions to construct event graphs, compresses them while preserving semantic structure, aligns multiple executions, identifies and categorizes execution divergences, and propagates these divergences through the application's symbol table to generate a function-level map of non-determinism. Rather than simply reporting that two executions differ, ND-Locate identifies the code regions responsible for those differences.

We evaluate ND-Locate on three representative MPI applications: MiniAMR, Quicksilver, and Enzo; and demonstrate that the functions identified by ND-Locate correspond to known sources of non-deterministic behavior. By localizing the origin of non-determinism, ND-Locate enables developers to focus debugging and optimization efforts on the code regions that most directly influence scientific reproducibility.

2 Methodology

Figure 2 summarizes the ND-Locate workflow. We repeatedly execute an MPI application under identical conditions and use ANACIN-X [1] to collect communication traces. Each execution yields an event graph capturing causal relationships among MPI operations. In our experiments, these traces produce graphs with up to 6.5 million vertices and 14.3 million edges across runs with up to 64 MPI ranks, motivating scalable graph compression and alignment.

Because event graphs may contain more than one million vertices and six million edges, ND-Locate performs semantics-preserving graph compression to reduce their size while retaining the communication patterns required for analysis. The compressed graphs are aligned across executions using learned vertex embeddings, allowing corresponding communication events to be matched despite execution differences. The resulting alignment is analyzed to identify and categorize regions of non-determinism.

Finally, these regions are propagated through the application's symbol table to construct a function graph that attributes execution-level non-determinism to application functions. This enables developers to identify not only that two executions differ, but also where non-determinism originates within the source code.

3 Experimental Validation

We evaluate ND-Locate using both controlled benchmarks and a production scientific application. The mini-applications *miniAMR* and *Quicksilver* are executed at 64 MPI ranks each, while the production *Enzo* experiment uses 16 MPI ranks and a problem configuration of 10 Mpc/h comoving box with a 16^3 root grid and

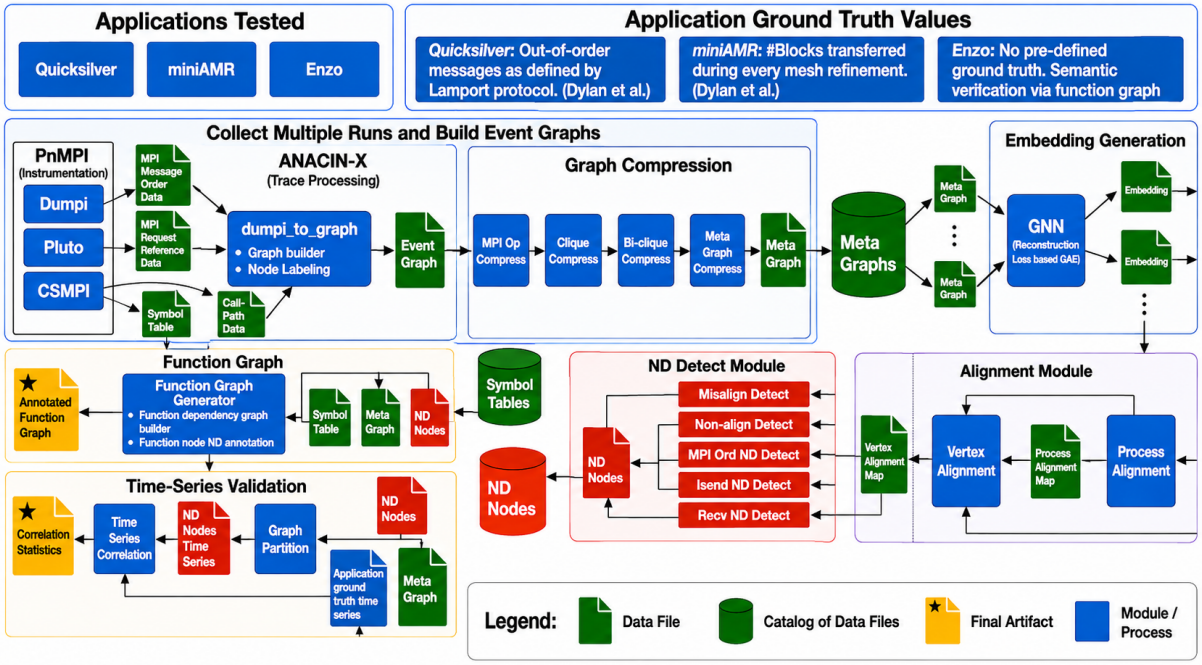


Figure 2: Overview of ND-Locate. Multiple MPI executions are converted into event graphs, compressed, aligned, analyzed for non-deterministic behavior, and mapped back to application functions.

Table 1: Validation against application-specific ground truth.

Application	ND Category	Pearson	dCor
miniAMR	Recv ND	**0.97**	**0.97**
Quicksilver	MPI Ord ND	**0.96**	**0.62**

16^3 dark-matter particles, refined adaptively over 3 levels (128^3 effective resolution), evolved from $z = 99$ to $z = 0$. We validate the accuracy of the framework on the mini-applications *miniAMR* and *Quicksilver*, where application-specific ground truth is available. We assess its usefulness on the production cosmological simulator *Enzo*, where the source of non-determinism is unknown a priori, via semantic analysis of the located functions.

For *miniAMR*, the ground truth is the number of block transfers at each refinement step, while for *Quicksilver* it is the number of out-of-order messages defined by the Lamport protocol. Table 1 summarizes the strongest correlations between the detected ND categories and the application-specific ground truth. In *miniAMR*, *Recv ND* achieves a Pearson correlation of 0.97 and a distance correlation (dCor) of 0.97, indicating an almost perfect match with the observed block transfers. In *Quicksilver*, *MPI Ord ND* achieves a Pearson correlation of 0.96 and a dCor of 0.62, correctly identifying the dominant source of communication non-determinism.

Unlike the benchmark applications, no ground-truth annotation exists for the source of non-determinism in *Enzo*. Instead, we assess whether the functions identified by ND-Locate are scientifically meaningful. The framework highlights communication and particle deposition routines, including `CommunicationReceiveHandler`, `grid::DepositBaryons`, `grid::AddOverlappingParticleMassField`, and `PrepareGravitatingMassField2a`, with ND scores ranging from 0.87 to 0.99. These routines perform non associative floating-point operations whose execution order directly influences particle accumulation and halo formation, providing strong evidence that ND-Locate successfully localizes the code regions responsible for changes in scientific outcomes.

4 Conclusions

ND-Locate precisely attributes execution-level non-determinism to application functions. Across benchmarks, it achieves Pearson correlations up to 0.97 with known sources and identifies the code regions causing output variability in the production cosmological simulator *Enzo*, demonstrating its accuracy for validation and its practicality for debugging large-scale scientific simulations.

Acknowledgments

The work is partially funded by NSF grants 1900765 and 1900888.

References

[1] Bell, P., Suarez, K., Chapp, D., Tan, N., Bhowmick, S., and Taufer, M. Anacin-x: A software framework for studying non-determinism in mpi applications. *Software Impacts 10* (2021), 100151.

[2] Chapp, D., Tan, N., Bhowmick, S., and Taufer, M. Identifying degree and sources of non-determinism in mpi applications via graph kernels. *IEEE Transactions on Parallel and Distributed Systems 32*, 12 (2021), 2936–2952.

[3] O'Shea, B. W., Bryan, G., Bordner, J., Norman, M. L., Abel, T., Harkness, R., and Kritsuk, A. Introducing enzo, an amr cosmology application, 2004.