Test of Time Award: Transparently Mixing Undo Logs and Software Reversibility for State Recovery in Optimistic PDES

Davide Cingolani
davide.cingolani@huawei.com
Huawei Technologies R&D
Cambridge, United Kingdom

Alessandro Pellegrini
a.pellegrini@ing.uniroma2.it
Tor Vergata University of Rome
Rome, Italy

Francesco Quaglia
francesco.quaglia@uniroma2.it
Tor Vergata University of Rome
Rome, Italy

ABSTRACT

The paper "*Transparently Mixing Undo Logs and Software Reversibility for State Recovery in Optimistic PDES*" introduced a seminal hybrid rollback technique that efficiently and transparently combines checkpointing and reverse computation methods through runtime-generated undo instructions. This short abstract, which accompanies the Test of Time Award received at PADS 2025, summarises the fundamental challenges presented in the original paper and reflects on its impact in the decade after its publication.

ACM Reference Format:
Davide Cingolani, Alessandro Pellegrini, and Francesco Quaglia. 2025. Test of Time Award: Transparently Mixing Undo Logs and Software Reversibility for State Recovery in Optimistic PDES. In *39th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '25), June 23–26, 2025, Santa Fe, NM, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3726301.3733243

1 INTRODUCTION

The article "*Transparently Mixing Undo Logs and Software Reversibility for State Recovery in Optimistic PDES*" [6] introduced an innovative approach to efficiently supporting rollback operations in optimistic Parallel Discrete Event Simulation (PDES). The main contribution lies in a hybrid strategy combining traditional checkpointing and reverse computation techniques [4] to restore previous simulation states after a causality violation. An expanded version of the article has been then published in [7].

The fundamental idea behind the work is that both checkpointing and reverse computation could be suboptimal for performing a rollback operation. Checkpointing tends to incur significant memory and CPU overhead, especially if full-state copies are saved frequently. Conversely, reverse computation could become expensive with complex event logic. Moreover, the modeller should typically provide reverse event-handlers, a process that requires additional manual effort and could be error-prone. To address limitations from both worlds, we proposed a hybrid recoverability architecture that relies on automatically-generated *undo code blocks*, i.e. a minimalistic form of reverse event-handlers that are generated on-the-fly while running forward events. One core peculiarity of such undo code blocks is that they revert state updates with no actual re-computation of the original state-location values. Rather, each instruction in an undo code block already embeds immediate operands for restoring the state-location value.

Overall, undo code blocks differ significantly from traditional reverse event-handlers because they are independent of event granularity, depending solely on the actual number of memory locations modified. They also differ from traditional undo logs because they do not require managing metadata to identify the correspondence between a saved state-location value and the corresponding memory location where to restore it at rollback time; as hinted, instructions in an undo code block are encoded directly as assembly operations that precisely revert state updates that have occurred at specific memory addresses. This is an interesting aspect, which is also linked to the effectiveness of the exploitation of the caching hierarchy, thanks to the reduction of the number of cache lines required for performing the reverse of the events' updates when a rollback occurs.

We also incorporated and readapted an analytical model borrowed from existing literature [9], to determine the optimal balance between checkpoint distance and the use of undo code blocks based on runtime dynamics. The experimental results showed that the hybrid technique provides substantial performance benefits, particularly in large-scale and high-load scenarios, typically those where optimistic PDES finds its most critical application.

2 A REFLECTION ON THE IMPACT

State restore is one of the core aspects in optimistic PDES, and relates to both performance aspects and transparency vs the application level code. Our original article has inspired additional research in both of these directions. Additionally, it has played a role in other fields of research. This section discusses a few works that have used our solution as a baseline.

Our software instrumentation technique—as mentioned, tailored for the runtime generation of undo code blocks—has been a reference for works where instrumentation has been exploited in orthogonal ways. In particular, we can mention solutions suited for offering reversibility of third-party libraries [8] or for tracking memory updates via the inclusion of Hardware-Level instrumentation [5] to reduce the CPU cycles needed for making the memory update reversible. Additionally, the solutions provided in [17, 20] exploit variations the baseline techniques characterizing our instrumentation process to build fully new mechanisms for the management of memory accesses, either for virtualizing a shared-memory platform on top of a distributed PDES system, or for supporting new typologies of simulation events no longer confined to a single simulation object of the PDES application. Also, the work in [16] provides the support for the exploitation of variations of the instrumentation technique to enable preemptability of simulation events,

SIGSIM-PADS '25, June 23–26, 2025, Santa Fe, NM, USA

which have instead been treated as atomic un-preemptable units in the previous literature.

At the same time, our proposal has been taken as a reference by works attempting to enlarge the possibility of reverting generic application code developed relying on specific programming languages [21]. Also, it has been considered when designing and evaluating orthogonal instrumentation schemes, not based on on-the-fly generation of reverse-handlers' instructions [22].

In general, the core reversibility technique we have presented has played a role in the design and development of the support for reversible programming, in particular when considering both concurrency in the operations [10] and program debugging [13]. Still in the field of PDES, the work in [1] presents a reversibility solution suited for a specific class of application models, which accounts for our proposal as a baseline reference.

Another area where our solution has had attention as a possible baseline is memoization. In particular, in [23] the authors discuss how a memoization process based on compile-time software instrumentation is somehow linked to our instrumentation scheme.

Our work has also been taken into account by publications where core techniques and methods in the area of (speculative) PDES are summarised, discussed and compared, thus forming the basis for additional research in the area [11, 18].

The core idea of mixing different algorithms in PDES to support runtime operations based on their cost has also been explored, with particular attention to synchronisation, in several works [11, 19]. This mixing approach has been moved from the software to the hardware level in [2, 15], where it has been shown that jointly using different classes of hardware can benefit the overall performance. At the same time, these works have highlighted that, when targeting heterogeneous architecture, methodologies different from instrumentation could be employed, such as model-driven engineering [3, 14], to support hybrid simulation.

Finally, real-time distributed applications have also been investigated in [12], taking our work into account as a means for effectiveness in the context of synchronisation.

REFERENCES

[1] Philipp Andelfinger, Jordan Ivanchev, David Eckhoff, Wentong Cai, and Alois Knoll. 2019. From effects to causes: Reversible simulation and reverse exploration of microscopic traffic models. In *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '19)*. ACM, New York, NY, USA, 173–184. https://doi.org/10.1145/3316480.3322891

[2] Philipp Andelfinger, Alessandro Pellegrini, and Romolo Marotta. 2024. Sampling policies for near-optimal device choice in parallel simulations on CPU/GPU platforms. In *Proceedings of the 28th International Symposium on Distributed Simulation and Real Time Applications (DS-RT '24)*. IEEE, Piscataway, NJ, USA, 101–109. https://doi.org/10.1109/ds-rt62209.2024.00023

[3] Simone Bauco, Romolo Marotta, and Alessandro Pellegrini. 2025. DESL: A Literate Programming Language Framework for Interoperable Parallel Discrete Event Simulation. In *Proceedings of the 2025 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '25)*. ACM, New York, NY, USA, 12. https://doi.org/10.1145/3726301.3728420

[4] Christopher D Carothers, Kalyan S Perumalla, and Richard M Fujimoto. 1999. Efficient Optimistic Parallel Simulations Using Reverse Computation. *ACM Transactions on Modeling and Computer Simulation* 9, 3 (July 1999), 224–253. https://doi.org/10.1145/347823.347828

[5] Davide Cingolani, Mauro Ianni, Alessandro Pellegrini, and Francesco Quaglia. 2016. Mixing hardware and software reversibility for speculative parallel discrete event simulation. In *Reversible Computation*. Springer International Publishing, Cham, 137–152. https://doi.org/10.1007/978-3-319-40578-0_9

[6] Davide Cingolani, Alessandro Pellegrini, and Francesco Quaglia. 2015. Transparently mixing undo logs and software reversibility for state recovery in optimistic PDES. In *Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '15)*. ACM, New York, NY, USA, 211–222. https://doi.org/10.1145/2769458.2769482

[7] Davide Cingolani, Alessandro Pellegrini, and Francesco Quaglia. 2017. Transparently mixing undo logs and software reversibility for state recovery in optimistic PDES. *ACM transactions on modeling and computer simulation: a publication of the Association for Computing Machinery* 27, 2 (April 2017), 1–26. https://doi.org/10.1145/3077583

[8] Davide Cingolani, Alessandro Pellegrini, Markus Schordan, Francesco Quaglia, and David R Jefferson. 2017. Dealing with reversibility of shared libraries in PDES. In *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '17)*. ACM, New York, NY, USA, 41–52. https://doi.org/10.1145/3064911.3064927

[9] Vittorio Cortellessa and Francesco Quaglia. 2000. Aggressiveness/Risk Effects Based Scheduling in Time Warp. In *Proceedings of the 2000 Winter Simulation Conference*, Jeffrey A Joines, Russel R Barton, Keebom Kang, and Paul A Fishwick (Eds.). IEEE, Piascatawy, NJ, USA, 409–417. https://doi.org/10.1109/WSC.2000.899746

[10] James Hoey and Irek Ulidowski. 2022. Reversing an imperative concurrent programming language. *Science of Computer Programming* 223 (2022), 102873. https://doi.org/10.1016/j.scico.2022.102873

[11] David R Jefferson and Peter D Barnes, Jr. 2022. Virtual time III, Part 1: Unified Virtual Time synchronization for parallel discrete event simulation. *ACM transactions on modeling and computer simulation: a publication of the Association for Computing Machinery* 32, 4 (Oct. 2022), 1–29. https://doi.org/10.1145/3505248

[12] Akio Kawabata, Bijoy Chand Chatterjee, Seydou Ba, and Eiji Oki. 2017. A real-time delay-sensitive communication approach based on distributed processing. *IEEE access: practical innovations, open solutions* 5 (2017), 20235–20248. https://doi.org/10.1109/access.2017.2758803

[13] Ivan Lanese. 2018. From reversible semantics to reversible debugging. In *Reversible Computation*. Springer International Publishing, Cham, 34–46. https://doi.org/10.1007/978-3-319-99498-7_2

[14] Romolo Marotta and Alessandro Pellegrini. 2024. Model-Driven Engineering for High-Performance Parallel Discrete Event Simulations on Heterogeneous Architectures. In *Proceedings of the 2024 Winter Simulation Conference (WSC '24)*, H Lam, E Azar, D Batur, S Gao, W Xie, S R Hunter, and M D Rossetti (Eds.). IEEE, Piscataway, NJ, USA, 2202–2213. https://doi.org/10.1109/WSC63780.2024.10838978

[15] Romolo Marotta, Alessandro Pellegrini, and Philipp Andelfinger. 2024. Follow the leader: Alternating CPU/GPU computations in PDES. In *Proceedings of the 38th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS'24)*. ACM, New York, NY, USA, 47–51. https://doi.org/10.1145/3615979.3656056

[16] Alessandro Pellegrini and Francesco Quaglia. 2017. A fine-grain time-sharing Time Warp system. *ACM transactions on modeling and computer simulation: a publication of the Association for Computing Machinery* 27, 2 (April 2017), 1–25. https://doi.org/10.1145/3013528

[17] Alessandro Pellegrini and Francesco Quaglia. 2019. Cross-state events: A new approach to parallel discrete event simulation and its speculative runtime support. *Journal of parallel and distributed computing* 132 (Oct. 2019), 48–68. https://doi.org/10.1016/j.jpdc.2019.05.003

[18] Kalyan Perumalla, Maximilian Bremer, Kevin Brown, Cy Chan, Stephan Eidenbenz, K Scott Hemmert, Adolfy Hoisie, Benjamin Newton, James Nutaro, Tomas Oppelstrup, Robert Ross, Markus Schordan, and Nathan Urban. 2022. *Computer Science Research Needs for Parallel Discrete Event Simulation (PDES)*. Technical Report 1855247. U.S. Department of Energy. https://doi.org/10.2172/1855247

[19] Andrea Piccione, Philipp Andelfinger, and Alessandro Pellegrini. 2023. Hybrid Speculative Synchronisation for Parallel Discrete Event Simulation. In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '23)*. Association for Computing Machinery, New York, NY, USA, 84–95. https://doi.org/10.1145/3573900.3591124

[20] Matteo Principe, Tommaso Tocci, Pierangelo Di Sanzo, Francesco Quaglia, and Alessandro Pellegrini. 2020. A distributed shared memory middleware for speculative Parallel Discrete Event Simulation. *ACM transactions on modeling and computer simulation: a publication of the Association for Computing Machinery* 30, 2 (April 2020), 1–26. https://doi.org/10.1145/3373335

[21] Markus Schordan, Tomas Oppelstrup, David Jefferson, and Peter D Barnes. 2018. Generation of Reversible C++ Code for Optimistic Parallel Discrete Event Simulation. *New Generation Computing* 36 (2018), 257–280. https://doi.org/10.1007/s00354-018-0038-2

[22] Markus Schordan, Tomas Oppelstrup, Michael Kirkedal Thomsen, and Robert Glück. 2020. Reversible languages and incremental state saving in optimistic parallel discrete event simulation. In *Reversible Computation: Extending Horizons of Computing*. Springer International Publishing, Cham, 187–207. https://doi.org/10.1007/978-3-030-47361-7_9

[23] Mirko Stoffers, Daniel Schemmel, Oscar Soria Dustmann, and Klaus Wehrle. 2018. On automated memoization in the field of simulation parameter studies. *ACM transactions on modeling and computer simulation: a publication of the Association for Computing Machinery* 28, 4 (Oct. 2018), 1–25. https://doi.org/10.1145/3186316