EvoChain: A Recovery Approach for Permissioned Blockchain Applications

Francisco Faria, Samih Eisa, David R. Matos and Miguel L. Pardal
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
{franciscofaria00, david.r.matos, miguel.pardal}@tecnico.ulisboa.pt
samih.eisa@inesc-id.pt

7 November 2025

Outline

- Introduction
- Background
- EvoChain
- Evaluation
- Conclusion

Outline

- Introduction
- Background
- EvoChain
- Evaluation
- Conclusion

Introduction (1)

- Blockchain technology introduced the possibility of maintaining a distributed ledger of transactions without the need for a central authority
- Immutable and tamper-resistant system, ensuring trust and transparency among participants
- Captured the attention of both academia and industry [1]:
  - Total venture capital funding in blockchain technologies reached 22.7 billion dollars by the third quarter of 2025

[1] Azam, F. (2025, August 4). Blockchain Statistics & Facts 2025. TekRevol. https://www.tekrevol.com/blogs/blockchain-statistics-facts/

Introduction (2)

- Immutability may be a hindrance in real-world applications
- Many systems require some degree of flexibility for data redaction or correction:
  - Rectify human errors
  - Intentional misconduct

Outline

- Introduction
- Background
- EvoChain
- Evaluation
- Conclusion

Background (1): Intrusion Recovery

Two common approaches:

- **Rollbacks:** reverting all the activity to a point where no damage exists
- **Compensating:** undoing committed or uncommitted transactions that affect others by doing traditional undos or applying special-purpose compensating transactions

Background (2): Blockchain

- Append-only distributed ledger that records transactions
- Managed by nodes that agree on a common transaction order
- Two main actors: clients and nodes
- Permissioned vs. Permissionless blockchains
- Managed by nodes that agree on a common transaction order

Background (3): Related Work

Bypassing Immutability

- Off-chain Storage [2]
- In-chain encryption [3]
- Pruning [4]
- Reversible Smart Contracts [5]

Removing Immutability

- Consensus-based [6]
- Chameleon-Hash [7]
- Meta-Transactions [8]

[2] J. Eberhardt and S. Tai, “On or off the blockchain? insights on off-chaining computation and data,” in Service-Oriented and Cloud Computing, F. De Paoli, S. Schulte, and E. Broch Johnsen, Eds. Cham: Springer International Publishing, 2017, pp. 3–15.
[3] “Encrypting on-chain data,” Sep 2021, [Accessed 10-Oct-2022]. [Online]. Available: https://research.csiro.au/blockchainpatterns/general-patterns/data-management-patterns/encrypting-on-chain-data/
[4] J. D. Bruce, “The Mini-Blockchain Scheme,” 2014.
[5] F. F. Martins, D. R. Matos, M. L. Pardal, and M. Correia, “Recoverable Token: Recovering from Intrusions against Digital Assets in Ethereum,” in 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA). Cambridge, MA, USA: IEEE, Nov. 2020, pp. 1–9.
[6] C. Jentzsch, S. It, C. Jentzsch, and S. It, “Decentralized Autonomous Organization To Automate Governance,” p. 31, Retrieved from https://lawofthelevel.lexblogplatformthree.com/wpcontent/uploads/sites/187/2017/07/WhitePaper-1.pdf.
[7] G. Ateniese, B. Magri, D. Venturi, and E. R. Andrade, “Redactable blockchain – or – rewriting history in bitcoin and friends,” 2017 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 111–126, 2017.
[8] I. Puddu, A. Dmitrienko, and S. Capkun, “µchain: How to forget without hard forks,” IACR Cryptol. ePrint Arch., vol. 2017, p. 106, 2017.

Outline

- Introduction
- Background
- **EvoChain**
- Evaluation
- Conclusion

EvoChain (1): Conceptual Proposal

- Transaction-based storage in the world state
  - Three states: pending, consolidated, or canceled
  - Three time fields: submission time, permanent state time, delay
- Direct control of the fabric's world state
- Dependency tracking between related transactions
- Apparent mutability introduced within the system
- Objects reconstructed by the View Generation component

EvoChain (2): Fabric Ledger

- World state stores the current values of ledger assets as key–value pairs
- Is derived from the immutable blockchain underneath

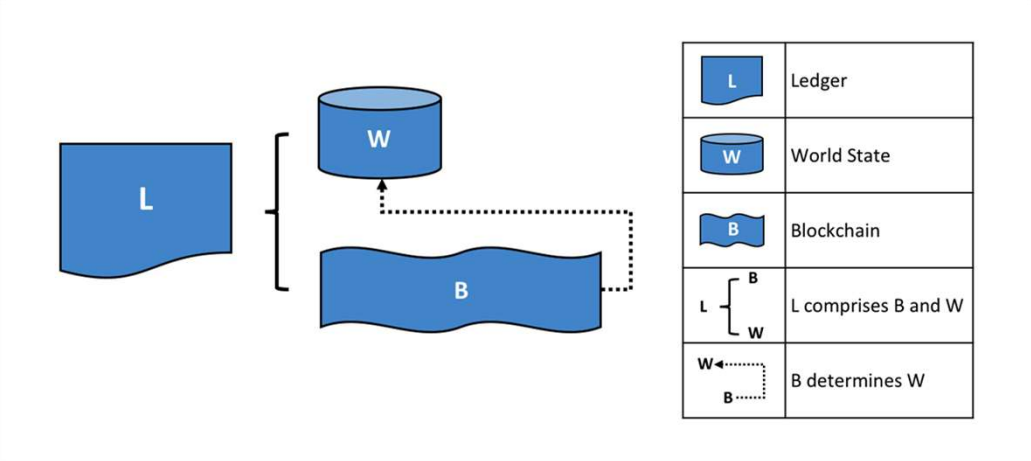

Figure 1: Fabric Ledger

EvoChain (3): Mutation Policies

- Set of rules that define if modifications to transactions are accepted.
- Transactions are first issued as pending and can end in one of two final states, consolidated or canceled:
- A transaction becomes consolidated in two possible ways:
  - Delay expiration: the configured time window has passed
  - Condition satisfaction: a predefined condition in the policy is met

EvoChain (4): View Generation

- The View Generation creates a specific data representation within the system
- Uses the high-level first-class entity transaction model to compose data
- Considers pending and consolidated transactions to calculate the final state of an object

Outline

- Introduction
- Background
- EvoChain
- **Evaluation**
- Conclusion

Evaluation (1): WineTracker

- Supply Chain Management Application based on selected models from relevant literature

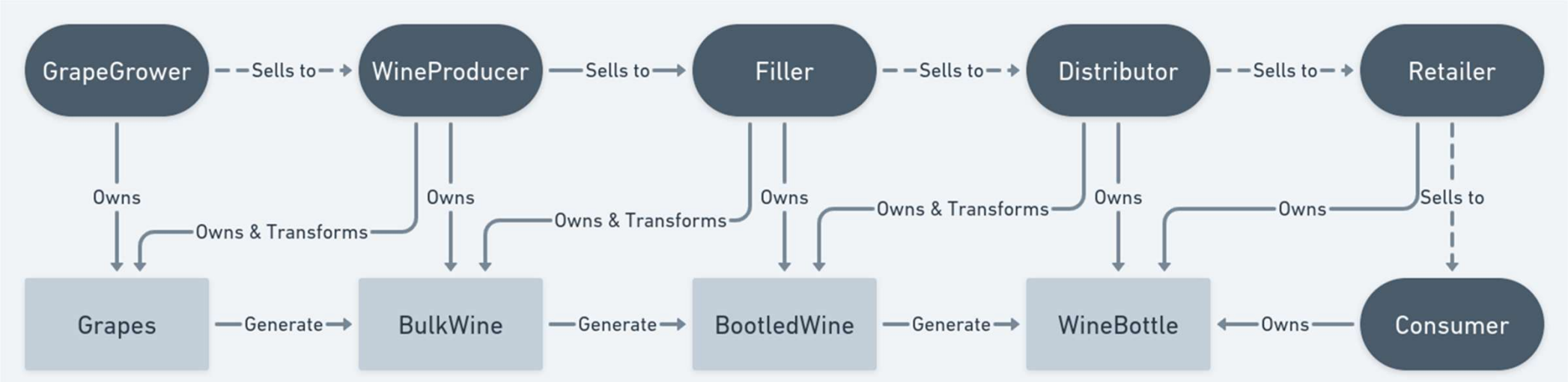

Figure 2: Network Model

Evaluation (2): EvoWineTracker

- Every “Sell” and “Transform” operation is stored as a transaction, instead of modifying specific objects.
- Transactions contain the new and the altered version of the objects.
- A relationship between objects is created.
- Reversal of operations is now possible.

Evaluation (3): Configuration

- Google Compute Engine instance, 16 vCPUs, 16GB of memory, 100GB Balanced Persistent Disk
- Four organizations, each represented by a single peer
- LevelDB used as world state database
- Raft-based consensus algorithm
- No private data and usage of range queries
- Transport Layer Security (TLS) encryption between entities
- Default Fabric policies and configurations

Evaluation (4): Performance Evaluation

- TC1: Evaluation of the performance of core functionalities, comparing both versions
- TC2: Performance of cancel transactions and their influence on core functionalities
- TC3: Evaluation of query transactions, considering that they are responsible for consolidating pending transactions with an expired delay

Evaluation (5): TC1

Table 5.1: TC1 WineTracker Functionality Metrics

Name	SendRate(TPS)	AvgLatency(s)	Throughput(TPS)	AverageMemory(MB)
createGrapes	353.0	26.14	189.5	376.75
sellGrapes	415.4	21.04	205.9	426.00
transformGrapes	409.6	12.98	239.8	432.38
sellBulk	409.4	10.15	258.0	439.38
transformBulk	411.1	12.36	242.1	450.00
labelBottles	445.1	40.04	139.2	476.38
sellDistributor	465.7	55.19	113.2	500.38
sellRetailer	463.7	56.33	111.1	506.75
sellConsumer	463.3	55.97	111.4	510.50
checkBottle	374.6	11.66	196.5	475.38

Table 5.2: TC1 EvoWineTracker Functionality Metrics

Name	SendRate(TPS)	AvgLatency(s)	Throughput(TPS)	AverageMemory(MB)
createGrapes	318.0	31.02	166.1	383.88
sellGrapes	421.5	23.64	201.9	420.00
transformGrapes	422.9	18.94	215.0	444.00
sellBulk	423.3	17.95	216.3	459.00
transformBulk	424.0	20.94	206.6	471.25
labelBottles	456.5	47.44	121.2	495.63
sellDistributor	446.1	57.15	109.0	506.25
sellRetailer	467.5	57.84	108.4	510.50
sellConsumer	474.8	56.85	108.5	516.00
checkBottle	414.6	24.32	185.3	488.88

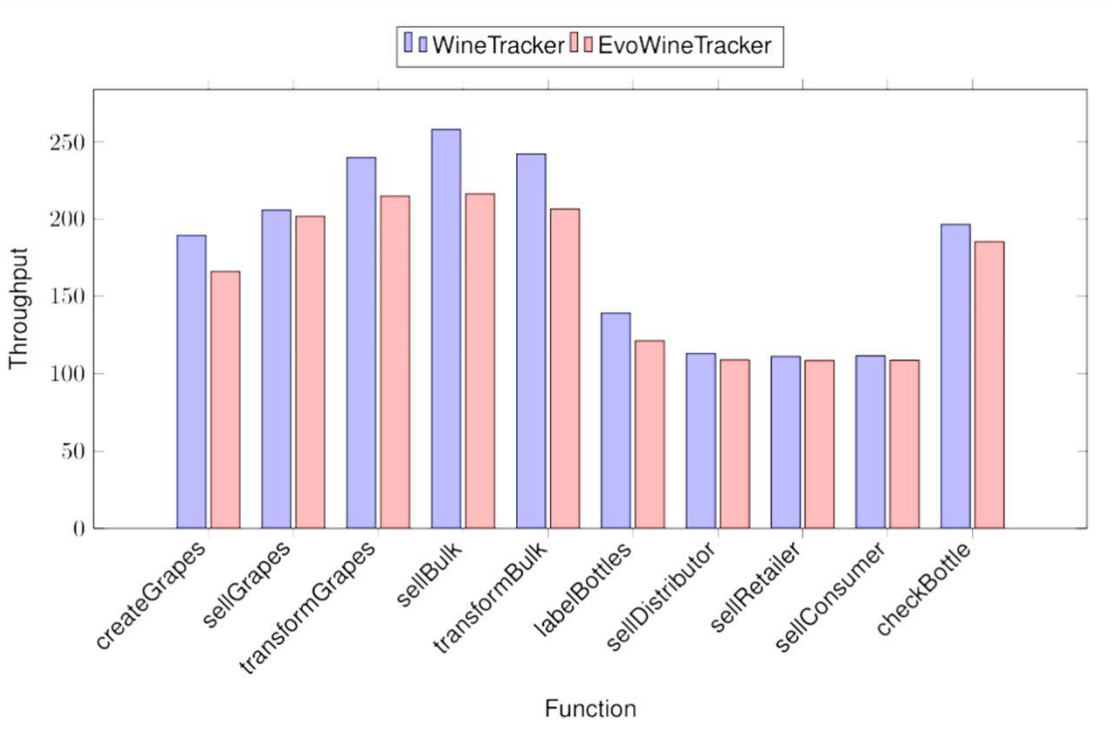

Figure 3: TC1 Throughput Comparison

Evaluation (6): TC2

Round	Name	Send Rate (TPS)	Avg Latency (s)	Throughput (TPS)
1	createGrapes	357.5	25.68	191.8
	sellGrapes	423.2	23.99	202.1
	transformGrapes	424.1	16.63	225.4
	cancelCreateGrapes	447.6	27.76	183.8
2	createGrapes	404.1	9.52	260.9
	sellGrapes	439.6	18.72	217.8
	transformGrapes	430.7	16.17	226.7
	cancelCreateGrapes	431.2	28.81	179.8
3	createGrapes	406.1	9.35	263.5
	sellGrapes	437.6	20.70	209.5
	transformGrapes	430.2	17.21	221.2
	cancelCreateGrapes	448.5	28.38	180.3
4	createGrapes	410.3	9.51	261.6
	sellGrapes	440.9	19.62	211.7
	transformGrapes	428.4	17.07	222.4
	cancelCreateGrapes	454.1	29.05	178.1
5	createGrapes	405.0	10.07	257.4
	sellGrapes	439.3	20.82	208.1
	transformGrapes	429.7	17.87	217.1
	cancelCreateGrapes	452.5	29.95	172.6

Table 5.3: TC2 Performance Data for Each Round

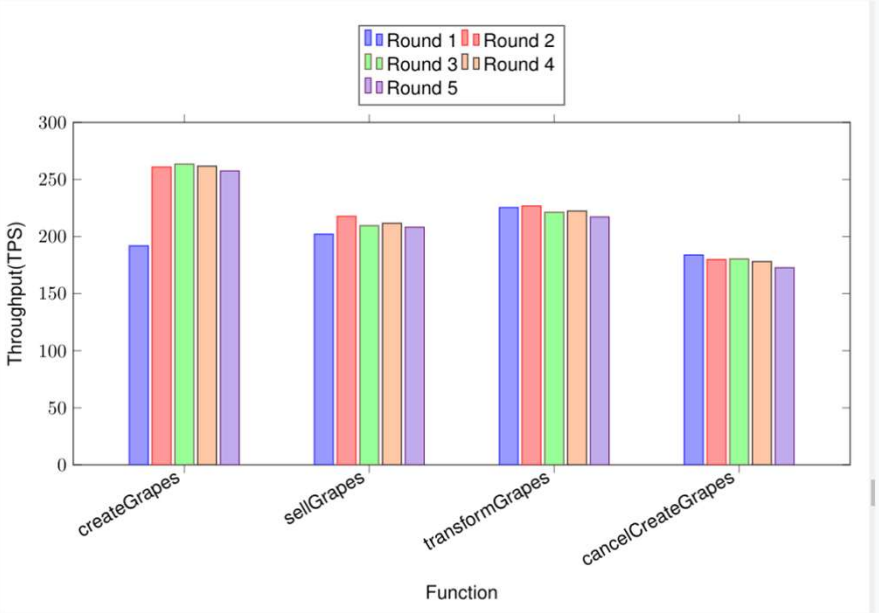

Figure 4: TC2 Throughput by Function and Round

Evaluation (7): TC3

Table 5.4: TC3 Consolation query operations

Name	SendRate (TPS)	AvgLatency (s)	Throughput (TPS)
getCreate	373.1	4.39	257.9
getSell	388.2	4.14	267.4
getTransform	402.0	4.31	270.5
getSellBulk	402.4	4.94	261.6

Table 5.5: TC3 No consolation query operations

Name	SendRate (TPS)	AvgLatency (s)	Throughput (TPS)
getCreate	403.8	3.07	294.6
getSell	402.7	3.55	287.7
getTransform	407.4	3.25	292.7
getSellBulk	407.3	3.47	288.9

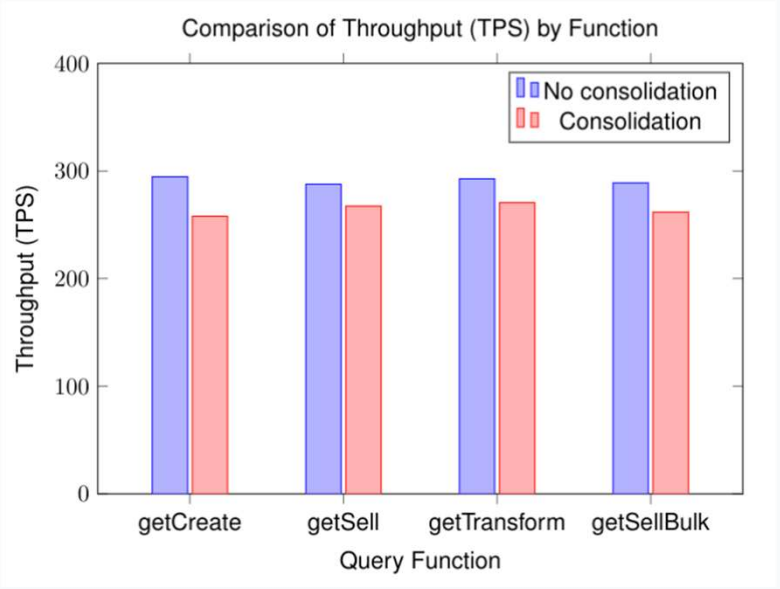

Figure 5: TC2 Throughput by Query Function

Outline

- Introduction
- Background
- EvoChain
- Use Case
- Evaluation
- Conclusion

Conclusion (1)

- The use of EvoChain as a base for chaincode development introduced a small decrease in throughput considering core functionalities and query transactions
- Transaction history enables higher-level rollback without compromising data integrity
- Inherent security properties regarding blockchain systems were not compromised

Conclusion (2) - Future Work

- Other applicational contexts can benefit from an optimization on data storage by exploring delta-based or partial object versions to reduce performance overhead
- Evaluate EvoChain in larger, more resource-intensive applications to assess scalability under longer mutation delays

Thank you / Obrigado

Work supported by national funds through Fundação para a Ciência e a Tecnologia, I.P. (FCT) under projects UID/50021/2025 and UID/PRR/50021/2025 (INESC-ID) and Project Blockchain.PT – Decentralize Portugal with Blockchain Agenda, (Project no 51), WP 1: Agriculture and Agri-food, Call no 02/C05-i01.01/2022, funded by the Portuguese Recovery and Resilience Program (PRR), The Portuguese Republic and The European Union (EU) under the framework of Next Generation EU Program.