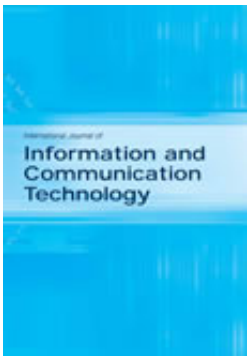
INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research



International Journal of Information and Communication Technology

ISSN online: 1741-8070 - ISSN print: 1466-6642
https://www.inderscience.com/ijict

Concurrent execution of transactions in blockchain: a framework for on-chain-off-chain nested contract processing

Yufang Xie, Guoqiong Liao, Yinxiang Lei

Article History:

Received:	27 October 2024
Last revised:	23 November 2024
Accepted:	25 November 2024
Published online:	20 January 2025

Concurrent execution of transactions in blockchain: a framework for on-chain-off-chain nested contract processing

Yufang Xie*

School of Information Management and Mathematics,
Jiangxi University of Finance and Economics,
Jiangxi, 330032, China
and
Jiangxi Science and Technology Normal University,
Jiangxi, 330038, China
Email: myResearch0312@163.com
*Corresponding author

Guoqiong Liao

Virtual Reality (VR) Modern Industry College,
Jiangxi University of Finance and Economics,
Jiangxi, 330032, China
Email: researchwork000@126.com

Yinxiang Lei

School of Information Management and Mathematics,
Jiangxi University of Finance and Economics,
Jiangxi, 330032, China
and
Jiangxi University of Traditional Chinese Medicine,
Jiangxi, 330004, China
Email: paperSubmission33@163.com

Abstract: With its natural benefits of decentralisation and immutability, blockchain technology has become rather popular in smart contracts, supply chains, and banking. It struggles greatly, nevertheless, in terms of scalability and transaction processing efficiency. Thus, in this regard, these studies present the execute-order-re-execute and validate (EOR) architecture to handle these problems. By means of off-chain execution, on-chain ordering, on-chain re-execution and verification phases, the EOR architecture maximises transaction processing, hence improving system performance and security. For nested contract concurrency it uses a two-phase locking technique, and for effective verification a lockchain architecture. Offering a significant means for extending blockchain uses, experimental results show a 40% boost in transaction processing efficiency, a 2.5% transaction abort rate, and enhanced system stability in high-conflict settings.

Keywords: blockchain; concurrency control; nested contracts; on-chain-off-chain.

Reference to this paper should be made as follows: Xie, Y., Liao, G. and Lei, Y. (2025) 'Concurrent execution of transactions in blockchain: a framework for on-chain-off-chain nested contract processing', *Int. J. Information and Communication Technology*, Vol. 26, No. 1, pp.73–88.

Biographical notes: Yufang Xie received Master's degree from the Adelaide University in 2011. Currently, she is a Doctoral student in the Jiangxi University of Finance and Economics, and works in Jiangxi Normal University of Science and Technology. Her research interests include blockchain and database.

Guoqiong Liao is a Professor at the Jiangxi University of Finance and Economics and the Director of the Virtual Reality (VR) Modern Industry Institute. His research interests include big data technologies, databases, data mining, meta-universe, blockchain and social networks.

Yinxiang Lei received her Master's degree from the Nanchang Hangkong University in 2013, and a Doctoral student in Jiangxi University of Finance and Economics. Her research interests are blockchain and database.

1 Introduction

Blockchain technology has grown to be a pillar in the domains of financial technology and distributed computing since Satoshi Nakamoto first proposed the idea in 2008 (Zhao et al., 2016). Blockchain's key benefit is that it offers a transparent, decentralised, tamper-proof data recording system. From financial transactions to smart contracts to supply chain management and government regulation, blockchain applications are growing and handling more and more sophisticated data as they do. But this expansion also brings certain difficulties, especially with relation to system scalability and transaction processing efficiency (Nasir et al., 2022).

Conventional blockchain systems, like Bitcoin and Ether, force all transactions to be carried out and validated on the chain, therefore restricting transaction flow. For instance, whereas Ether is just competent of roughly 30 transactions per second, the Bitcoin network can handle almost seven transactions per second. These constraints have mostly limited the use of blockchain technology in a greater spectrum of contexts (Buterin, 2014).

Scholars have put up some ideas to overcome these restrictions. Among the off-chain scaling alternatives, Dwivedi et al. (2021) suggested Plasma, a sidechain technology-based scaling method that transfers part of the transaction processing off-chain, hence increasing efficiency. Furthermore greatly enhancing the transaction processing capacity of the Bitcoin network, Zabka et al. (2022) introduced the lightning network, a system using stateful channels to enable instantaneous off-chain payments.

Regarding on-chain scaling methods, scholars have investigated dynamic block resizing and sharding approaches. For instance, Xu et al. (2017) suggested a dynamic block sizing system to fit the several network requirements. Aiming to increase the scalability of the network, Liu et al. (2022) later closely examined blockchain sharding methods.

These methods somewhat solve the performance issue, but they sometimes call very sophisticated network coordination systems or compromise of some of the decentralised characteristics of blockchain. We thus propose an execute-order-re-execute and verify (EOR) transaction processing framework that enables high concurrency, i.e., EOR, in order to overcome the issues of ineffective transaction processing and poor system scalability in blockchain systems. By use of blockchain transaction processing mechanism optimisation, the EOR framework separates transaction processing into three phases: off-chain execution, on-chain sequencing, and on-chain re-execution and verification, thereby improving system performance and security.

The contributions and innovations of this paper include:

1 Increases the system's scalability and lessens the load on the chain's nodes' storage and computing. Through assigning the data storage and part of the transaction processing to the third-party SPs under the chain, the EOR framework essentially lowers the resource consumption of the nodes, so improving the scalability of the system.

2 Concurrency control techniques help to increase node processing transaction efficiency. The EOR structure helps to arrange SPs and transaction activity into several groups for concurrent running. SP nodes of the NCEP-2PL system may simultaneously handle several transactions including concurrent execution of nested contracts.

3 It guarantees system security even when transaction processing's efficiency is raised. Given the unreliable character of the off-chain execution environment, the EOR framework guarantees the security of the blockchain by means of fast chain verification using the NCVP-LC protocol, so ensuring that the transaction execution results are accurate without depending on further hardware or sophisticated cryptography.

4 Increases the transaction processing efficiency in highly conflict environments. By means of conflict identification and resolution in the on-chain sequencing phase, the EOR framework re-executes transactions aborted owing to conflicts, so mitigating the issue of transactions not being able to be committed in high conflict environments and so improving the stability and efficiency of the system.

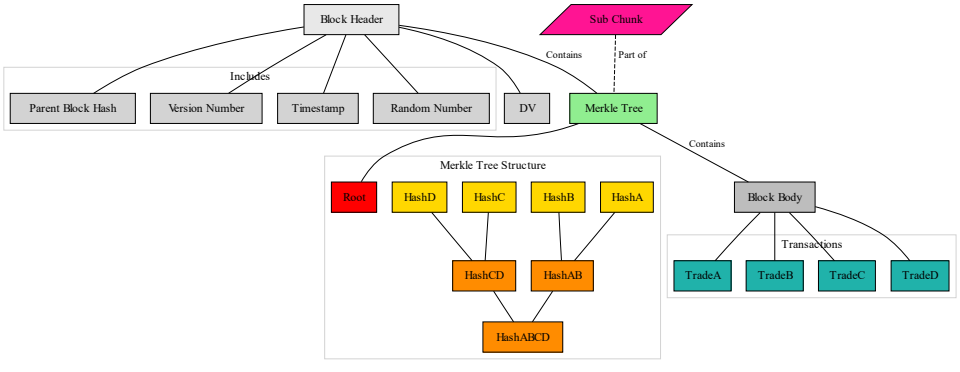
2 Relevant technologies

2.1 Blockchain

Fundamentally a distributed ledger system running on a peer-to-peer network, blockchain helps to enable decentralised data storage (Sarmah, 2018). Every node in the network under this system keeps a whole history of transactions, which, once found and included to the blockchain, become quite difficult to change, therefore ensuring data immutability and traceability. Fundamentally, the decentralised character of blockchain eliminates the dependence on centralised power since no single node can manage the whole network and therefore increases the system's resistance to attack and censorship. Furthermore, blockchain's open character lets any user check and audit transaction records, so boosting the system's whole legitimacy.

The blockchain's data architecture consists of a chain formed from a sequence of connected blocks arranged chronologically (Wei et al., 2022). Every block has not only transaction data from a given period but also the hash value of the one before it; this arrangement creates the special chain structure of the blockchain. As Figure 1 shows, every block in a blockchain is made of numerous fundamental elements guaranteeing network integrity and security.

Figure 1 A model of the blockchain (see online version for colours)



This picture shows the architecture of a blockchain block, stressing the block header, transaction list, and cryptographic hash connecting to the next block, therefore offering a visual assistance to help one to grasp the composition and purpose of every block.

- Parent block hash: Forming a chain structure, the hash value of the prior block, which links the current block to the blockchain, links before the current block, encryption of the whole blockchain data generates this hash, where H stands for the hash function.

$$Parent\ block\ hash = H(previous\ block) \tag{1}$$

- Version number: The version number of the blockchain protocol helps to guarantee that every node in the network recognises the format and guidelines of the block data. Usually, this value is a set integer meant to differentiate across several blockchain protocols.

$$Version = protocol\ version \tag{2}$$

- Timestamp: Where t is the Unix timestamp at the time the block was formed, the timestamp helps ascertain the sequence of the blocks in the blockchain and offers the temporal background of the transaction.

$$Timestamp = t \tag{3}$$

- Difficulty value: Where D is a dynamically changed value depending on the network arithmetic, the difficulty value is a network parameter that regulates the complexity of the proof of work algorithm and guarantees that the rate of production of new blocks corresponds the expected rate of the network.

$$Difficulty = D \tag{4}$$

- Merkle tree: By means of a hash function, which enables nodes in the network to rapidly verify the existence and integrity of a transaction, so preserving the compactness of the block data, Merkle tree is a tree data structure that organises multiple transactions within a block into a tree structure and combines them into a single root hash, the Merkle root (Diván and Sanchez-Reynoso, 2021).

$$Merkle\ root\ hash = H\left(H\left(H\ldots H\left(Transation\ Hash_1\right)\|\ldots\right)\right) \tag{5}$$

- Proof of work (PoW): Where target is a value less than or equal to 2,256 and difficulty is a difficulty parameter set by the network. Under PoW blockchain systems, like Bitcoin, miners must identify a specific value (nonce) such that, when added to this value, the hash value of the block header satisfies a given difficulty criteria. Target is a value less than or equal to 2,256; difficulty is a parameter the network sets.

  Miners change the nonce value in the block header as part of the PoW process to generate a hash that satisfies the network difficulty criterion. If the network calls for a hash with at least four leading zeroes, for example, miners would run through nonce values until they come across one that generates a hash like '0000abcde…'.

$$Target = Difficulty \times 2^{256} \tag{6}$$

  Blockchain technology is based on these elements taken together, so it is a dependable, open and quick distributed ledger system.

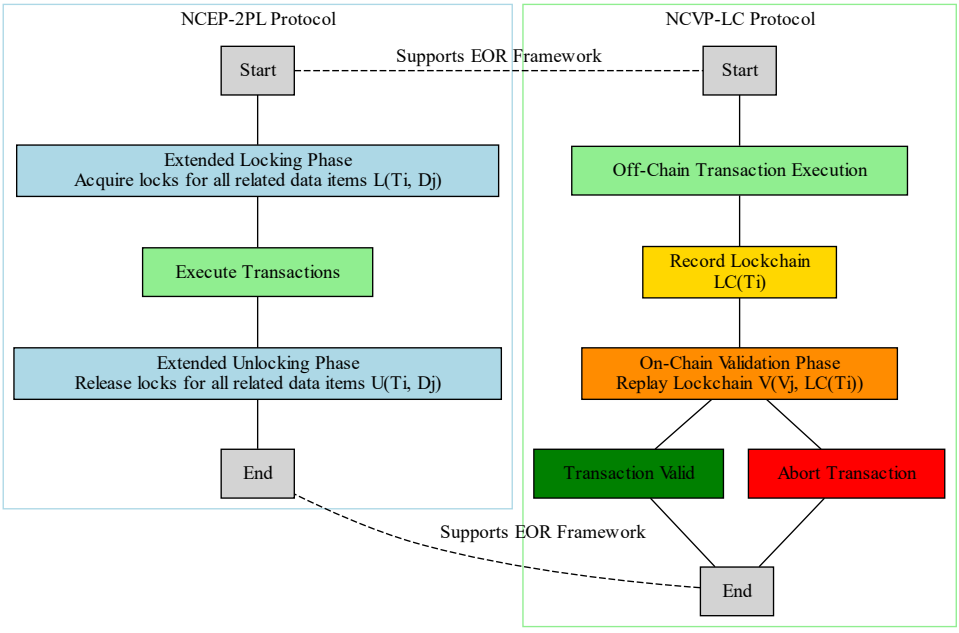
2.2 Concurrency control

Concurrency control techniques are required in blockchain systems to preserve integrity and data consistency since several nodes could process transactions concurrently (Meng et al., 2021). Concurrency control aims to guarantee that there is no inconsistency or data loss and that the state of the system is constant even in cases of concurrent multiple transactions. Ensuring data consistency and integrity depends mostly on concurrency control (Paik et al., 2019). While the tamperability of blockchain demands that once a transaction is confirmed, the distributed character of blockchain means that each node must independently confirm the authenticity of a transaction. Concurrency control policies must thereby guarantee the atomicity and durability of transactions, which presents special difficulties. In this context, as depicted in Figure 2, we propose two novel concurrency control protocols: the NCVP-LC protocol and the NCEP-2PL protocol.

To control transaction concurrency the NCEP-2PL system uses a two-phase locking mechanism. Transactions in the first phase lock the data items they must access for reading or writing. After their activities, transactions release these locks in the second phase therefore enabling subsequent transactions to access the data. This system guarantees data consistency by making sure no two transactions concurrently change the same data item.

The NCVP-LC system logs and checks transaction execution using a lockchain architecture. The path of every transaction is noted as a chain of locks and data operations. The transaction's execution on-chain is then confirmed using this chain, therefore guaranteeing accurate and safe off-chain execution results.

Figure 2 Structure of NCEP-2PL and NCVP-LC protocol (see online version for colours)



Because of their ordered, linked character, knowledge graphs are especially suited for revealing intricate connections inside data. By modelling entities and their interactions as nodes and edges respectively, they enable the application of graph analytics to find patterns and insights maybe not clear in other data structures. For our EOR system, this functionality is absolutely essential since it allows intelligent analysis and processing of enormous volumes of blockchain data.

Blockchain technology is based on these elements taken together, so it is a dependable, open, quick distributed ledger system.

Designed to increase transaction processing efficiency in blockchain systems, the NCEP-2PL protocol – a two-phase locking-based method for the execution of layered contracts – aims to By means of a two-phase locking mechanism, hence extending the conventional two-phase locking protocol to support concurrent execution of nested transactions, the protocol permits concurrent execution of transactions and nested contracts in an off-chain context. Every transaction in NCEP-2PL has to acquire locks on all pertinent data items before execution and release these locks following completion of the transaction. Two phases comprise this process: the first is the prolonged locking phase, in which the transaction gains the required locks; the second is the extended unlocking phase, in which case the locks are released following data processing completion. In this sense, the NCEP-2PL system guarantees the atomicity and consistency of transactions, therefore enhancing the concurrent processing capacity of the system.

Designed to rapidly validate transactions carried out under the chain on the chain, the NCVP-LC protocol is a nested contract validation mechanism grounded on the lockchain structure. Like a chained table, where each data access operation is noted as a node and the nodes are connected to one another by the order of locks, the protocol records data access and locking information using the lockchain structure. Replaying the lockchain

allows nodes on the chain to deterministically execute transactions concurrently during the verification phase, therefore confirming the accuracy of the execution along the chain. This method guarantees blockchain security while avoiding the necessity to depend on more hardware or intricate cryptographic methods.

We can propose the following formula to better specify how these two protocols operate and explain the locking and unlocking mechanism of a transaction in the NCEP-2PL protocol. Imagine T_i must access a data item D_j in a transaction. Transaction T_i must get a lock on D_j, denoted $L(T_i, D_j)$, in the extended locking phase. Transaction T_i releases the lock after operation on D_j, represented as $U(T_i, D_j)$, in the extended unlocking phase. One can show this method as follows:

$$L(T_i, D_j) \rightarrow O_p(T_i, D_j) \rightarrow U(T_i, D_j) \tag{7}$$

where L stands for the locking action; O_p for the operation on data item D_j; U for the unlocking action.

We can explain the transaction verification procedure for the NCVP-LC protocol by means of this formula. Assume that after a transaction T_i is carried out under the chain; its matching lockchain is $LC(T_i)$. Replaying $LC(T_i)$, represented as $V(V_j, LC(T_i))$, the verification node V_j ensures the accuracy of transaction T_i in the on-chain verification phase. Should the replay go through, transaction T_i's execution outcome is accurate; else, transaction T_i must be cancelled. One may depict this procedure as follows:

$$V(V_j, LC(T_i)) \rightarrow \boxed{\textit{The execution result of the transaction } T_i \textit{ is correct}} \tag{8}$$

$$V(V_j, LC(T_i)) \rightarrow \boxed{\begin{array}{l}\textit{The execution result of the transaction TI is wrong}\\ \textit{and it needs to be aborted}\end{array}} \tag{9}$$

These two protocols ensure atomicity and permanence of transactions, therefore enabling the EOR framework to enhance the concurrency processing capability and security of the blockchain system. These protocols' design considers the special qualities of blockchain systems, such distributed authentication and immutability, therefore allowing the EOR framework to efficiently handle problems with concurrency in blockchain systems.

2.3 Nested contracts

A breakthrough idea in blockchain technology, smart contracts let us enforce contractual terms transparently and decentralised without middlemen (Mik, 2017). Blockchain's smart contract capability adds still another level of application. Smart contracts are basically digital contracts based on pre-defined criteria stored on the blockchain, self-executing digital contracts with terms automatically performed upon certain conditions satisfied.

A smart contract's working concept can be reduced as a conditional triggering mechanism whereby every clause of the contract is expressed logically. The smart contract acts automatically in line with the pre-defined conditions when they are satisfied. One can depict this by the following logical flow:

$$Contract = \{C | C = (P, S, M)\} \tag{10}$$

where smart *Contract* is *C*; *Predicate* is *P*; *State* is *S*; performed action is *M*. The smart contract automatically carries out the related action *M* when the condition *P* holds, therefore altering the state *S* of the contract.

Local and state variables can both be included into smart contracts. Local variables exist just during contract execution; state variables are kept on the blockchain. Usually, changes to state variables call for triggered transactions:

$$State\ variable = New\ value \tag{11}$$

Smart contracts' fundamental ability to automate and decentralise contract execution in many application contexts helps to define their working concept (Zheng et al., 2020). From the financial industry to supply chain management, identity verification, and many other sectors, the application scope of smart contracts is growing as blockchain technology develops; however, a single smart contract is usually inadequate to manage all the business logic. By now the idea of nested contracts emerged. A nested contract is a smart contract capable of calling another smart contract, hence creating a chain of contract calls. Implementing sophisticated business logic and creating modular blockchain applications calls for this approach (Six et al., 2022). Nestled contracts – which enhance transaction processing complexity – invite mutual invocations between contracts, hence enhancing their flexibility and scalability even while they complicate matters. One can depict the execution of a hierarchical contract by means of a recursive formula.

$$Nested\ contract\ execution\left(C_1, C_2, \ldots, C_n\right) = C_1\left(C_2\left(\ldots C_n \ldots\right)\right) \tag{12}$$

This formula uses C_1, C_2, …, C_n to symbolise the smart contracts called upon in turn. Every contract may set off the following one, creating an invocation chain. Although this recursive execution approach offers a strong instrument for developing sophisticated business logic, it also presents significant difficulties particularly in concurrent execution and transaction management.

3 EOR transaction processing framework

3.1 Overview of the EOR framework

Node sort to maximise performance and scalability, the EOR framework divides system nodes into two primary categories: on-chain and off-chain. While on-chain nodes function in a lightweight way, storing just the required little amount of data and concentrating on transaction sequencing and validation, off-chain nodes assume the main responsibility for transaction processing and data storage. While increasing general stability and efficiency, this architecture distributes the heavy computation and storage chores to the off-chain nodes, therefore relieving the pressure on the on-chain nodes and guaranteeing the decentralised character of the system. The framework separates nodes specifically into four distinct roles with various obligations.

This can be represented by the following code block:

```
// Example of a simple smart contract for demonstration purposes
contract SimpleContract {
    // State variable
    uint256 public data;
    // Function to update data
    function updateData(uint256 _data) public {
        data = _data;
    }
    // Function to retrieve data
    function getData() public view returns (uint256) {
    return data;
    }
}
```

Client nodes connect with the system via the blockchain network and send tasks to be completed; they are in charge of starting transaction demands.

Found under the chain, service provider (SPs) nodes handle associated data storage and specific transaction processing activities. Having great computing and storage capacity, they are the primary workers used in storage and execution.

These on-chain nodes serve to gather transaction execution results from SPs and use particular algorithms to arrange these transactions in readiness for next consensus procedures.

Also found on the chain, validation and re-execution nodes (VR nodes) are in charge of verifying the accuracy of the transaction results provided by SPs and, should necessary ensure that all transactions follow the consensus of the blockchain by means of re-execution of the transactions.

Data organisation under the EOR structure, data security and accessibility are guaranteed while the data storage approach seeks to lower the load on the on-chain nodes. The framework uses an off-chain storage mechanism that moves vast volumes of data and transaction execution information to off-chain nodes in order to reach this aim. SPs, off-chain nodes, are in charge of keeping the whole blockchain's history including transaction records, smart contract codes, and state of affairs.

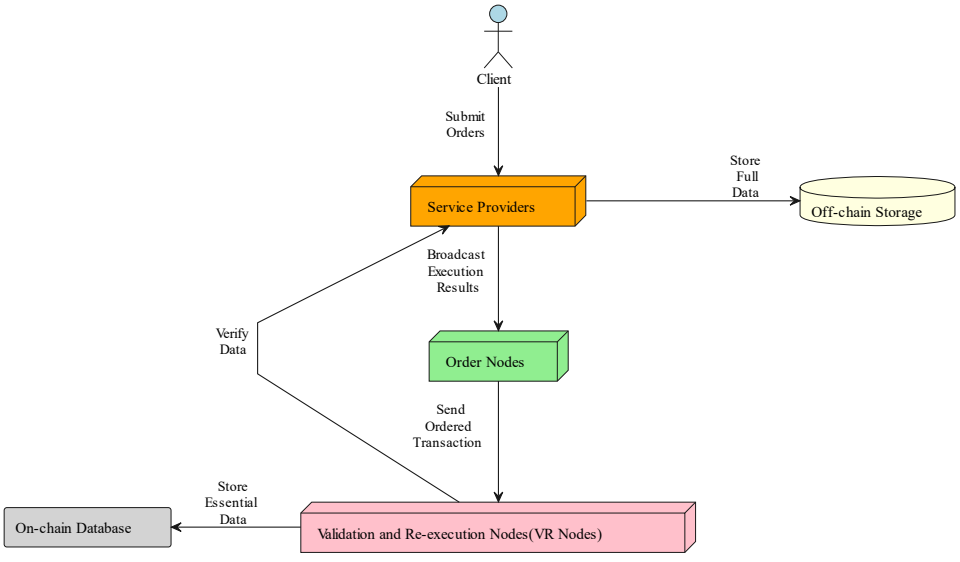
Conversely, on-chain nodes have a lightweight function by keeping just required data straight connected to the consensus process, including block header information and the latest state tree. Enough information in the block header – the hash of the previous block, the consensus proof, the root hash of the transaction tree, the root hash of the receipt tree – allows one to verify the integrity of the block and the validity of the transaction without storing the data of the whole block.

Furthermore, the on-chain node might decide to keep the hashes of every block header in order to raise data verifiability. By means of Merkle tree proofs, a client can thus query the on-chain node to get the root hash of the data and confirm the accuracy of the data query results offered by the off-chain node. This approach lets the system greatly lower the on-chain nodes' storage needs without compromising security, hence increasing the whole network's scalability.

Well-designed node categorisation and optimal data storage techniques in the EOR framework offer a steady and effective working environment. By lowering the load on the nodes on the chain, these solutions not only guarantee the decentralised character of the network but also offer the potential of rapid and safe transaction processing. This drives the EOR framework to provide a set of transaction actions to guarantee the effective running of the whole blockchain system.

From the start of the transaction until its final recognition and storage, the EOR transaction processing system controls the lifetime of a transaction by means of a well defined process, see Figure 3.

Figure 3 The EOR framework (see online version for colours)



The following are the key steps in transaction processing in the framework:

1 Transaction proposal: First building transaction requests, the client node transmits them down the chain to the SP nodes. These calls provide all the required transaction data – sender, receiver, transaction parameters.

2 Transaction execution: Based on current state data, SPs down the chain answer transaction requests and carry out those actions. Included with state modifications and resultant data, the execution's outcomes are recorded and ready for commit.

3 Transaction sorting: Transactions executed have to be arranged on the chain. Receiving the execution results from the SPs, the sorting node sorts these transactions with a particular method to guarantee their correct sequence.

4 Transaction VR: Verifying the sorted transactions falls to the re-execution and validation (RV) node on the chain. Should a flaw arise during the validation process, the RV node will re-run these transactions to guarantee the consistency and accuracy of every transaction.

5 Once the transactions are confirmed, they are lastly validated and kept on the blockchain. While the on-chain nodes record important summary information to keep a lightweight and high performance network, the off-chain nodes are in charge of storing thorough transaction data and history.

By means of end-to-end encryption, our off-chain storage system guarantees data protection. Before transmission, every data packet is encrypted on the off-chain node and only deciphered by the appropriate on-chain node, therefore maintaining data confidentiality throughout travel. Maintaining an index of data locations on the on-chain ledger helps to ensure accessibility by enabling effective data retrieval as required for validation or consensus procedures.

3.2 Assessment indicators

This experiment seeks to examine the EOR framework's performance throughout processing blockchain transactions holistically. We consider transaction processing time, system resource use, transaction abort rate, and performance comparison of the EOR framework with current methods. We will create a lot of blockchain transactions and test the stability and effectiveness of the EOR framework on these transactions by simulating several network situations and user actions.

Especially helpful in handling high-dimensional data typically seen in blockchain applications, sparse representations are a data representation method whereby only the non-zero elements of a dataset are saved, therefore greatly lowering the memory needs.

1 Transaction processing time: From the moment a transaction is committed to until it is accepted and noted on the blockchain, transaction processing time is the average time (Helo and Shamsuzzoha, 2020). This measure directly shows the blockchain's transaction processing transaction efficiency. Fast transaction processing time is absolutely vital for user experience and system performance in a highly concurrent environment. Ti is the processing time of the i^{th} transaction; T_{avg} is the average transaction processing time; n is the total number of transactions.

$$T_{avg} = \frac{1}{n}\sum_{i=1}^{n} T_i \tag{13}$$

2 Concurrent execution efficiency: Usually expressed by transaction throughput and transaction abort rate, concurrent execution efficiency is the capacity of a system to effectively perform several transactions concurrently. High concurrent execution efficiency indicates that the system may execute more transactions without compromising transaction integrity and system stability (Rajwar and Goodman, 2003). This is computed with spit as the number of effectively handled transactions per unit of time, and n is the total number of transactions; T_{total} is the time required to handle all the transactions.

$$Throughput = \frac{n}{T_{total}} \tag{14}$$

3 Concurrent execution efficiency: The percentage of transactions that, for a variety of reasons – conflicts, mistakes, etc. – fail to complete properly is known as the transaction abort rate. A low transaction abort rate is a main measure of the

efficiency of concurrency management systems since it lowers wasted resources and raises user satisfaction (Lam et al., 2002). Calculated using the following formula – where m is the total number of aborted transactions – it yields.

$$Transaction\ abort\ rate = \frac{m}{n} \times 100\% \tag{15}$$

Regarding concurrent execution efficiency, consider a system whereby several transactions start concurrently. Within a specific period – say one minute – we monitor the number of these transactions that are effectively handled. The system shows great concurrent transaction handling efficiency if 100 transactions are started and 95 are effectively handled in the minute. This example shows how concurrent execution efficiency may be practically used to evaluate the capacity of the system to handle several transactions at once without sacrificing integrity or speed of processing.

4 Experimental results and analyses

4.1 Experimental setup

We used a basic abstract contract as an experimental benchmark since standardised nested contracts are not easily found right now. On the blockchain, the contract code reads and writes data as well as additional logic unrelated to reading and writing data, such some corporate logic or computation. The experiment aims to compare the acceleration efficiency of the protocol for transaction execution since clearly the EOR framework suggested in this research only affects the reading and writing of data, not other code logic. We thus present smart contracts with varying complexity and specify several concurrency levels: 10%, 30%, 50%, 70% and 90%.

To meet the demands of extremely parallel processing, the experiments were carried out on high performance servers fitted with Intel Xeon Gold 6140 CPUs and NVIDIA Tesla V100 GPUs. We chose Linux operating system and apply the most recent version of Hyperledger Fabric as the blockchain platform. We particularly set the network environment to incorporate high latency and high throughput scenarios to test the performance of the EOR framework under these situations, therefore simulating various network states.

4.2 Experimental content

We used a basic abstract contract as an experimental benchmark since standardised nested contracts are not easily found right now. On the blockchain, the contract code reads and writes data as well as additional logic unrelated to reading and writing data, such some corporate logic or computation. The experiment aims to compare the acceleration efficiency of the protocol for transaction execution since clearly the EOR framework suggested in this research only affects the reading and writing of data, not other code logic. We thus present smart contracts with varying complexity and specify several concurrency levels: 10%, 30%, 50%, 70% and 90%.

4.2.1 Baseline testing phase

At this point, we want to set a performance baseline for next analysis of concurrent execution on system performance. We provide a control environment whereby only one transaction at a time is handled and no other concurrent transaction disturbs the outcome. Every transaction is timed; the time spent from transaction commit to execution completion is noted. Several times the procedure should be repeated to guarantee data dependability and stability. As a standard for later concurrency testing, find the average processing time of every single transaction. This stage allowed us to precisely assess the EOR framework's processing capability free from conflicting factors influencing it.

Table 1 shows the baseline test findings; the average EOR framework processing a single transaction without concurrent processing takes 4.8 seconds. This covers the whole transaction cycle – from commit to execution completion. This period represents the ideal case processing power of the EOR architecture since there is no interference from other concurrent transactions. It is noteworthy that every transaction carried out successfully within the allocated period without any errors or timeouts, therefore demonstrating the stability and dependability of the EOR system in handling individual transactions.

Table 1 Results of baseline tests

Transaction number	*Submission time*	*Finish time*	*Processing time(s)*
1	2024-05-23 10:00	2024-05-23 10:05	5
2	2024-05-23 10:06	2024-05-23 10:10	4
…	…	…	…
4	2024-05-23 11:30	2024-05-23 11:35	5

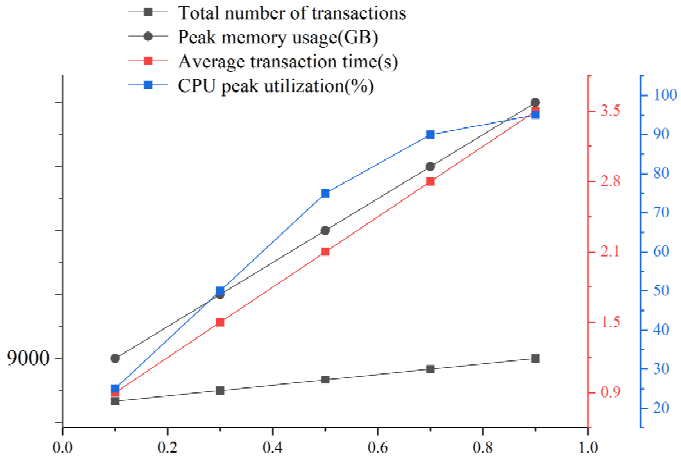
Note: Average processing time: 4.8 seconds.

4.2.2 Concurrent execution of test phases

We will progressively raise the concurrent level of transaction processing following baseline establishment. Concurrency levels for this phase will be 10% at 30% at 50% at 70% at 90% respectively. We will construct the amount of transactions matching that concurrency level at every level concurrently applying the EOR structure. We will note the overall level processing times as well as the time needed to complete all of the transactions. To evaluate the EOR framework's resource needs under a heavy load, we will also closely track the CPU and memory use of the system. This will enable us to better grasp under various loads the scalability and resource efficiency of the EOR system.

Figure 4 shows the concurrent execution test; as concurrency rises, the average transaction processing time progressively rises as a result of growing system resource competition – that is, CPU and memory. The system resource use is rather low at 10% concurrency level and the transaction processing time is almost equivalent to the level of the baseline test. The CPU and memory use rises dramatically as the concurrency level rises, mirroring the system's resource needs as it runs more concurrent transactions. The EOR framework performs well in resource management and transaction scheduling since, despite the increase in processing time, the system does not suffer any notable performance bottlenecks and the framework is still able to process transactions effectively.

Figure 4 Concurrent execution results (see online version for colours)



4.2.3 Transaction suspension rate test phase

One of the main markers of a concurrency control system's performance is its transaction abort rate. In this phase, we will focus especially on conflicts-related transaction aborts. We will replicate a high-conflict situation in which concurrency issues could cause half of the transactions to need to abort and retry. At every concurrency level, we will document the percentage of transactions aborted owing to conflicts; we will then investigate how these aborts affect the general performance. Evaluating the success of the EOR framework's conflict resolving approach depends on this stage.

Figure 5 Transaction abort rate test results (see online version for colours)

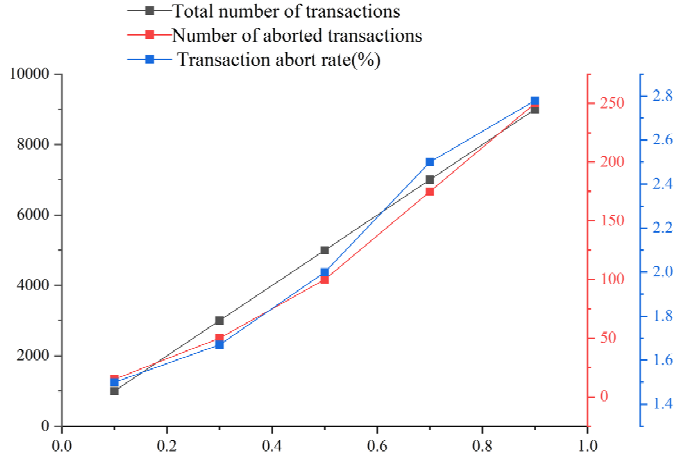


Figure 5 shows the outcomes of the transaction abort rate test, therefore demonstrating the capacity of the EOR framework to manage transactions in a highly conflict environment. Low transaction abort rates at reduced concurrency levels suggest that most of the transactions can be carried out successfully. The transaction abort rate rises as the

concurrency level rises; yet, with a high concurrency level of 90%, the abort rate is still under control at 2.78%, a rather low level. This outcome shows that most transactions can be carried out successfully since the concurrency management mechanism of the EOR framework can efficiently manage conflicts. Although in a highly concurrent environment some conflicts are unavoidable, the EOR framework uses efficient conflict resolution techniques to reduce the effect of transaction abortions.

Combining the foregoing experimental results, the EOR framework exhibits outstanding performance in managing rather concurrent blockchain transactions. In high-conflict situations, it not only fast processes transactions but also keeps a low transaction abort rate. These characteristics make the EOR framework a potential answer for blockchain application situations needing great efficiency and high throughput.

5 Conclusions

In this work, we design a two-stage locking-based nested contract execution protocol (NCEP-2PL) allowing concurrent execution of transactions and nested contracts in an off-chain environment, so extending the conventional two-stage locking protocol to support concurrent execution of nested transactions by means of a two-stage locking mechanism. Furthermore based on the lockchain architecture, we suggest the nested contract verification protocol (NCVP-LC) for quick on-chain transaction off-chain verification. These protocols help the EOR framework to guarantee atomicity and consistency of transactions and thereby enhance the concurrent processing capability of the system.

We design a two-stage locking-based nested contract execution protocol (NCEP-2PL), so extending the conventional two-stage locking protocol to support concurrent execution of nested transactions by means of a two-stage locking mechanism in an off-chain environment. Moreover depending on the lockchain architecture, we propose the nested contract verification protocol (NCVP-LC) for rapid off-chain transaction verification. These protocols improve the concurrent processing capacity of the system by guaranteeing atomicity and consistency of transactions, hence strengthening the EOR framework.

By means of experimental evaluations, we show that the EOR framework may significantly increase the transaction processing efficiency, lower the transaction abort rate, and strengthen system stability and efficiency in high-conflict surroundings. These findings indicate that the EOR framework offers a fresh approach for blockchain transaction processing, which is crucial for advancing the deployment of blockchain technology in a larger spectrum of conditions.

The research in this publication also has several restrictions even if the EOR architecture exhibits outstanding performance in tests. First of all, the EOR framework's experimental evaluation has been mostly carried out in simulated environments and has not yet been widely applied and verified in actual blockchain systems. Second, more study is necessary on the performance and stability of the EOR architecture in very high concurrency situations. The following two elements will be the main emphasis of next projects:

1 Further enhance the EOR architecture to raise its stability and performance under very high concurrency conditions.

2 Investigate the scalability and usability of EOR frameworks on several blockchain systems and application contexts and so provide more general support for the evolution of blockchain technology.

References

Buterin, V. (2014) 'A next-generation smart contract and decentralized application platform', *White Paper*, Vol. 3 No. 37, pp.2–1.

Diván, M.J. and Sanchez-Reynoso, M.L. (2021) 'Metadata-based measurements transmission verified by a Merkle tree', *Knowledge-based Systems*, Vol. 219, p.106871.

Dwivedi, V., Norta, A., Wulf, A. et al. (2021) 'A formal specification smart-contract language for legally binding decentralized autonomous organizations', *IEEE Access*, Vol. 9, pp.76069–76082.

Helo, P. and Shamsuzzoha, A. (2020) 'Real-time supply chain – a blockchain architecture for project deliveries', *Robotics and Computer-Integrated Manufacturing*, Vol. 63, p.101909.

Lam, K-Y., Kuo, T-W., Kao, B. et al. (2002) 'Evaluation of concurrency control strategies for mixed soft real-time database systems', *Information Systems*, Vol. 27, No. 2, pp.123–149.

Liu, Y., Liu, J., Salles, M.A.V. et al. (2022) 'Building blocks of sharding blockchain systems: concepts, approaches, and open problems', *Computer Science Review*, Vol. 46, p.100513.

Meng, T., Zhao, Y., Wolter, K. et al. (2021) 'On consortium blockchain consistency: a queueing network model approach', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 32, No. 6, pp.1369–1382.

Mik, E. (2017) 'Smart contracts: terminology, technical limitations and real world complexity', *Law, Innovation and Technology*, Vol. 9 No. 2, pp.269–300.

Nasir, M.H., Arshad, J., Khan, M.M. et al. (2022) 'Scalable blockchains – a systematic review', *Future Generation Computer Systems*, Vol. 126, pp.136–162.

Paik, H-Y., Xu, X., Bandara, H.D. et al. (2019) 'Analysis of data management in blockchain-based systems: from architecture to governance', *IEEE Access*, Vol. 7, pp.186091–186107.

Rajwar, R. and Goodman, J. (2003) 'Transactional execution: toward reliable, high-performance multithreading', *IEEE Micro*, Vol. 23, No. 6, pp.117–125.

Sarmah, S.S. (2018) 'Understanding blockchain technology', *Computer Science and Engineering*, Vol. 8, No. 2, pp.23–29.

Six, N., Herbaut, N. and Salinesi, C. (2022) 'Blockchain software patterns for the design of decentralized applications: a systematic literature review', *Blockchain: Research and Applications*, Vol. 3, No. 2, p.100061.

Wei, Q., Li, B., Chang, W. et al. (2022) 'A survey of blockchain data management systems', *ACM Transactions on Embedded Computing Systems (TECS)*, Vol. 21, No. 3, pp.1–28.

Xu, P., Corman, F., Peng, Q. et al. (2017) 'A train rescheduling model integrating speed management during disruptions of high-speed traffic under a quasi-moving block system', *Transportation Research Part B: Methodological*, Vol. 104, pp.638–666.

Zabka, P., Foerster, K-T., Schmid, S. et al. (2022) 'Empirical evaluation of nodes and channels of the lightning network', *Pervasive and Mobile Computing*, Vol. 83, p.101584.

Zhao, J.L., Fan, S. and Yan, J. (2016) 'Overview of business innovations and research opportunities in blockchain and introduction to the special issue', *Financial Innovation*, Vol. 2, pp.1–7.

Zheng, Z., Xie, S., Dai, H-N. et al. (2020) 'An overview on smart contracts: challenges, advances and platforms', *Future Generation Computer Systems*, Vol. 105, pp.475–491.