Journal of Combinatorial Mathematics
and Combinatorial Computing
www.combinatorialpress.com/jcmcc

Combinatorial Press

Research on Consistency Assurance Mechanism of Distributed Database Based on Blockchain Technology

Xingyan Shi[1,] ✉

[1] *Faculty of Information Engineering, Henan Vocational College of Agriculture, Zhengzhou, Henan, 451450, China*

ABSTRACT

With the rapid development of blockchain technology, consistency assurance of distributed database has become one of the key issues. In this paper, a blockchain distributed database consistency assurance mechanism based on the practical Byzantine fault tolerance (Rpbft) algorithm and its improved algorithm is studied in depth.The RPBFT algorithm combines the RSA algorithm and the PBFT consensus algorithm, and then performs the signature operation after message encryption in order to increase the system security. Aiming at the shortcomings of the master node selection mechanism of the original algorithm and the RPBFT algorithm, a master node selection mechanism that includes the time factor is proposed, which introduces the role of the recording node, so that the waiting time of the node can be adjusted dynamically. Meanwhile the algorithm changes the conditions of view switching and reduces the system consumption. Through simulation experiments to verify the performance of this paper's R-PBFT algorithm and OmniLedger and RapidChain two programs in the same network conditions, this paper's algorithm compared to the comparison algorithm can be more effective in guaranteeing the consistency of the distributed database, when the number of slices is 20, the transaction latency time is 13s, 25s lower than that of RapidChain and OmniLedger, respectively. When the number of shards is 20, the transaction delay time is lower than that of RapidChain and OmniLedger by 13s and 25s respectively, which provides strong support for the application of blockchain technology in the field of distributed database.

Keywords: Blockchain, PBFT consensus algorithm, RSA algorithm, Practical Byzantine Fault Tolerance Algorithm, Consistency Guarantee

1. Introduction

In recent years, with the rapid development of Internet applications, distributed systems are getting more and more attention. In distributed systems, distributed database is a very important component,

✉ Corresponding author.
E-mail address: 13849029903@163.com (X. Shi).
Received 05 November 2024; Revised 20 December 2024; Accepted 15 March 2025; Published Online 16 April 2025.
DOI: 10.61091/jcmcc127b-161

which can support massive data storage and fast data query [1]. This, in turn, creates the following three advantages at the application level:

First, the elimination of spatial restrictions, thus making the construction of the database more convenient, especially in the process of multi-data center construction to form a synergy of network resources, and has a certain contribution to data node transmission, data security, and data integrity [2-5].

Secondly, the distributed database improves the reliability of network information transmission, due to its use of multi-node, multi-storage approach to the integration of the construction, in the case of regional or part of the pass-through network failure or equipment failure can still be used in the form of a "bypass" to complete the relevant information retrieval and transmission, thus making the database based on the overall network stability has been greatly improved. The overall network stability has been greatly improved [6-8].

Third, in the construction of distributed database system, due to the database architecture of data and load distribution in multiple nodes, different sub-libraries exist independently, which makes the expansion of the database only need to expand the sub-libraries, in the process of expansion does not affect the overall application of the database [9-10].

However, the application of distributed databases also faces a series of challenges, such as the data consistency problem. The data consistency problem is one of the biggest challenges faced by distributed databases. In distributed systems, there are two data consistency models, strong consistency and weak consistency [11]. Strong consistency means that the system guarantees that the data of all nodes are consistent at any moment during the update process. Under the strong consistency model, each update operation is synchronized to all nodes before returning the result, under this model, strong data consistency is guaranteed, but it also brings a huge system overhead [12-14]. Weak consistency means that during the update process, the system cannot guarantee that the data of all nodes are consistent at any moment. Generally, the weak consistency model is more suitable for large-scale distributed systems because it reduces the system overhead, but at the same time it sacrifices the data consistency [15-16]. In distributed systems, each node may update data at different points in time, and such updates may lead to data inconsistency [17]. Therefore, how to ensure data consistency among nodes becomes an important issue.

This paper first introduces the principle of PBFT consensus algorithm and its application defects in blockchain environment, and establishes the research strategy of optimizing blockchain PBFT consensus algorithm using RSA cryptographic algorithm. In addition, due to the limitations of the view switching mechanism of the PBFT algorithm and the Rpbft algorithm, the probability of selecting a malicious node is extremely high. Therefore, a fixed waiting time is added to the algorithm to monitor the behavior of each node at regular intervals. Finally, using the simulation test platform Hyperledger Fabric v1.0, the R-PBFT optimization strategy of this paper and the RapidChain algorithm and OmniLedger algorithm are examined in comparison tests under different network sizes and node numbers.

2. Blockchain related knowledge

2.1. Blockchain Definition and Structure

The descriptive concept of blockchain first appeared in the Bitcoin whitepaper, but the whitepaper did not give a definition and concept of blockchain in an objective sense. Although the pace of development of blockchain technology in various fields has been accelerating in recent years, its technology is not perfect and many variants exist. As a result, there is currently no unified definition.

Blockchain technology is often referred to as distributed ledger [18] technology. Although it is referred to as ledger technology, its essence is only an abstract concept of a database consisting of a block structure. After understanding it as a database in a distributed environment, it is possible to

break the limitations of financial application scenarios and find other applicable areas to apply blockchain technology to all scenarios where global and historical data need to be recorded.

Generally speaking, the overall architecture of blockchain system can be divided into six layers: data layer, network layer, consensus layer, incentive layer, contract layer and application layer. The infrastructure model of blockchain technology is shown in Figure 1.

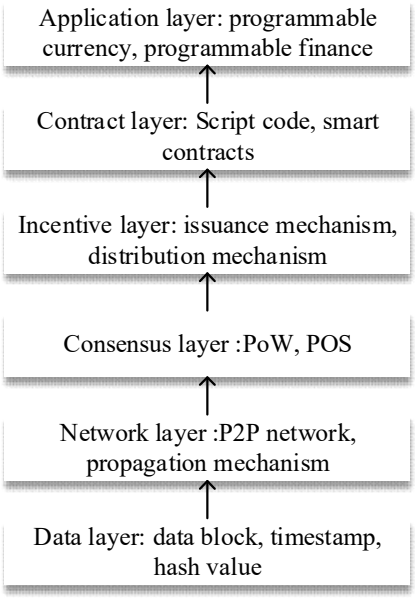

Fig. 1. Blockchain infrastructure

2.2. Smart Contracts

Smart contracts are a system of protocols that enable automatic execution. It enables blockchain to evolve from the blockchain 1.0 period of monetary transactions to the blockchain 2.0 period in the financial field and the 3.0 period of programmable society. Smart contracts are characterized by security and reliability and can be applied in a variety of fields such as digital payment, financial trade, Internet of Things, smart manufacturing, and healthcare.

Smart contracts are credible and efficient. When the predetermined conditions are triggered, the corresponding contract terms can be automatically enforced, and the entire generation and execution process is tamper-proof, secure and efficient. Due to the decentralized nature of the blockchain, smart contracts do not require the participation of a centralized authority, and can be run on all nodes of the whole network at the same time to ensure the traceability and irreversibility of transactions.

A smart contract consists of a computer program deployed on the blockchain, which ensures safe and efficient transactions, expands the functions of the blockchain, and enriches the application scenarios of the blockchain. After the smart contract code is written according to the needs of individual or multiple users, it is broadcasted through peer-to-peer network and deposited into the blockchain.

2.3. Consensus mechanisms

The consensus mechanism [19] determines the scalability and security of the blockchain system. Blockchain system is equivalent to a distributed database ledger, where nodes in a distributed environment jointly participate in decision-making to generate and maintain all transaction data. Consensus mechanism to reach consensus is the process of reaching a unified agreement on the state of blockchain system nodes in a decentralized way, which solves the problem of blockchain system

bookkeeping rights and the consistency of the ledger data, and ensures that the data of the nodes in the network are verified and updated.

The protocol rules of the consensus mechanism protect the state update of the distributed ledger of the blockchain system and realize the data transaction in the untrustworthy network environment. Consensus mechanism plays an important role in guaranteeing data consistency, fault tolerance and reliability of blockchain system.

The consensus mechanism mainly includes strong consistency consensus algorithm and final consistency consensus algorithm. Paxos and Raft algorithms are widely used in distributed systems, and they solve the consensus problem when there is a fault error in the distributed system. However, in distributed networks, in addition to considering fault errors, malicious behavior of nodes i.e. Byzantine errors also need to be considered.

PBFT algorithm solves the consensus problem in distributed systems in the presence of fault errors and malicious behavior of nodes.

2.3.1. Paxos and Raft Consensus Algorithms. Paxos is a consensus algorithm for solving non-Byzantine problems. The algorithm divides distributed nodes into three types of roles: proposer, acceptor, and learner.

Paxos implements a two-phase submission: a preparation phase and a confirmation phase.

During the operation of the Raft algorithm, there are two main phases, Leader election and synchronization log:

1) Lcader Election: at the beginning each distributed node is a Follower and an Election Timeout is set. When the countdown timer of the node ends the state will change to Candidate and sends an election request to other Follower. As long as half of the nodes vote for the most recent election phase, the node's state changes from Candidate to Leader and sends a Heartbeat to the other Follower for a certain period of time to maintain its state.

2) synchronization log: Raft in practical application scenarios is more in the distributed node data storage consistency, and to meet a few nodes out of order, the other nodes so that the existing data to reach a smooth consensus. Log (the occurrence of various events recorded) and through the Leader one-way forced synchronization to the Follower nodes.

2.3.2. PBFT consensus algorithm. The PBFT algorithm [20] can tolerate the security and activity of the system in case of no more than $\left\lfloor \frac{|R|-1}{3} \right\rfloor$ node being evil in the distributed network.

Suppose there are $|R|$ consensus nodes in the blockchain network. Each node is numbered sequentially from $\{0,1,....,|R|-1\}$. The master and slave nodes are collectively referred to as consensus nodes. The basic process of PBFT algorithm is as follows:

1) Firstly, the selection of master node p is carried out and one master node corresponds to one view v.

$$|R|=3f+1 \tag{1}$$

$$p=v \bmod |R| \tag{2}$$

2) In view v, the client sends the request command $<REQUEST,o,t,c>$ to the master node p, which receives the request and broadcasts it to other slave nodes.

3) All consensus nodes return the processed request $<REPLY,v,n,t,c,r,i>$ to the client, i with node number, and the client takes at least $f+1$ the same processed reply as the final result.

Among them, the master node broadcasting process is divided into three phases: pre-preparation phase, preparation phase, and confirmation phase:

1) Pre-preparation phase: the master node assigns proposal number n to the received client request and pre-preparation message $<PRE\_PREPARE,v,n,digest>,message>$, is multicast to

other slave nodes. Where digest is the summary of the message.

2) Preparation phase: after receiving the pre-prepared message sent by the master node, the slave node starts to verify the legitimacy of the message digest and ensures that the proposal number n satisfies the waterline take. If the verification passes, it multicasts the preparation message $< PREPARE, v, n, digest, k >$ to the whole network, collects the preparation messages from other nodes, and writes the pre-preparation message and the preparation message to the message mnemonic; otherwise, it does nothing. A consensus node enters the ready state if it collects $2f+1$ ready messages (including itself) that are the same as the preready message.

$$h \le n \le H \tag{3}$$

3) Acknowledgement phase: a consensus node that enters the ready state multicasts acknowledgement message $< COMMIT, v, n, D(m), k >$ to the whole network and collects acknowledgement messages from other nodes. If the consensus node collects $2f+1$ confirmation messages that are the same as the pre-prepared messages (including itself), the node enters the submit state. That is, it executes the client request and finally returns the execution result $< REPLY, v, t, c, r, k >$ to the client, where r represents the final execution result.

A view of this consensus mechanism v consists of a master node and a number of slave nodes, which always listen to the messages broadcast by the master node. The master node may be a dishonest node, so the slave nodes actively check the legitimacy of the messages sent by the master node and detect whether the master node is down through the timeout mechanism. If the slave node finds that the current master node is down, it triggers the View Change protocol.

PBFT consensus algorithm is the first BFT-like algorithm used in Hyperledger Fabric open source platform, which can well solve the problem of blockchain chain fork, but the author found that the algorithm also has certain limitations in the process of specific project development. The following analyzes the 2 deficiencies that exist in the PBFT consensus algorithm:

1) All consensus nodes have equal chances to act as a master node, i.e., a master node in View is elected by the master node selection formula $p = v \bmod |R|$. If the newly elected master node is a non-honest node, it will result in continuous master selection, and the external service of the whole blockchain system during the master node selection period will be greatly reduced or even unable to provide normal external service.

2) PBFT consensus algorithm transmits messages through message broadcasting, in addition to Pre-prepare, Prepare, Commit three-phase broadcasting, but also includes View Change protocol, Timeout mechanism, etc., so much message broadcasting exacerbates the consumption of network bandwidth. In a blockchain network with the number of consensus nodes N, the number of messages to be delivered to complete a three-phase broadcast is Z, then the relationship between the two is shown below:

$$Z = 2N^2 - 2N \tag{4}$$

3. Improved PBFT consensus algorithm based on RSA

In this chapter, firstly, the encryption delay and decryption delay experiments are carried out on the RSA encryption algorithm, and in the case of verifying that the degree of its delay is up to the standard, the PBFT consensus algorithm is optimized through the introduction of the RSA encryption algorithm, so as to achieve the purpose of improving the PBFT consensus algorithm without being restricted by its own limitations.

3.1. RSA algorithm

RSA encryption algorithm [21] is an asymmetric cryptographic algorithm. Compared with symmetric cryptographic algorithms, asymmetric cryptographic algorithms have higher security, this is because,

when encrypting and decrypting the same piece of information, the former will use a different key, while the latter will use the same key for encryption and decryption, and this situation will probably be selected as, as long as the attacker has a key, he or she can easily modify and destroy the information.

Step1: Randomly select two prime numbers q and p that are large enough and $q \neq p$.

Step2: Substitute q and p, where the prime numbers also need to satisfy Euler's formula $\varphi(n) = (p-1)(q-1)$.

$$n = p \times q \tag{5}$$

Step3: Pick the integer $e(1 < e < \varphi(n))$, where e is the public key index.

$$gcd(e, \varphi(n)) = 1 \tag{6}$$

Step4: Then derive the private key index d.

$$e \times d = 1 \bmod(\varphi(n)) \tag{7}$$

Step5: Calculate the public key $KU = \{e, n\}$ and private key $KS = \{d, n\}$.

The public key derived from the above steps is $\{e, n\}$ and the private key is $\{d, n\}$, where $d, p, q, \varphi(n)$ is not published in the network. Before the message is encrypted, it is also necessary to convert the plaintext to an integer m (m satisfies the condition: $m < n$). The plaintext is then encrypted into a ciphertext c.

$$c = E(m) = m^e \bmod n \tag{8}$$

If you want to decrypt ciphertext c into plaintext m.

$$m = D(c) = c^d \bmod n \tag{9}$$

3.2. Improving the Consistency Consensus Process

Each stage of node communication in the PBFT algorithm uses its own private key for encryption and public key for decryption to sign the data, which is used to determine whether the data has been maliciously tampered with. At the same time, the algorithm also needs to go through multiple stages of verification to ensure the security and reliability of the data, making it subject to certain limitations.

In the face of the limitations of the PBFT algorithm signature method, this paper proposes a signature method combined with RSA encryption algorithm. In this method, the data is first encrypted with RSA encryption, i.e., the sender first uses the receiver's public key to encrypt the data, and then signs the ciphertext with its own private key.

PBFT has high communication consumption in both Prepare and Commit phases, which makes it limited in practical applications. In order to reduce the communication complexity of PBFT, we will reduce the consistency processing in PBFT by improving the signature method in PBFT to reduce its communication complexity.RPBFT is based on the PBFT algorithm improved and optimized in this way.

RPBFT algorithm improves the PBFT algorithm according to the cryptographic properties of RSA encryption algorithm, changes the signature method of PBFT algorithm and also optimizes it for the consistency consensus process. In this paper, the specific implementation process of RPBFT algorithm will be explained. The specific flow chart is shown in Figure 2.

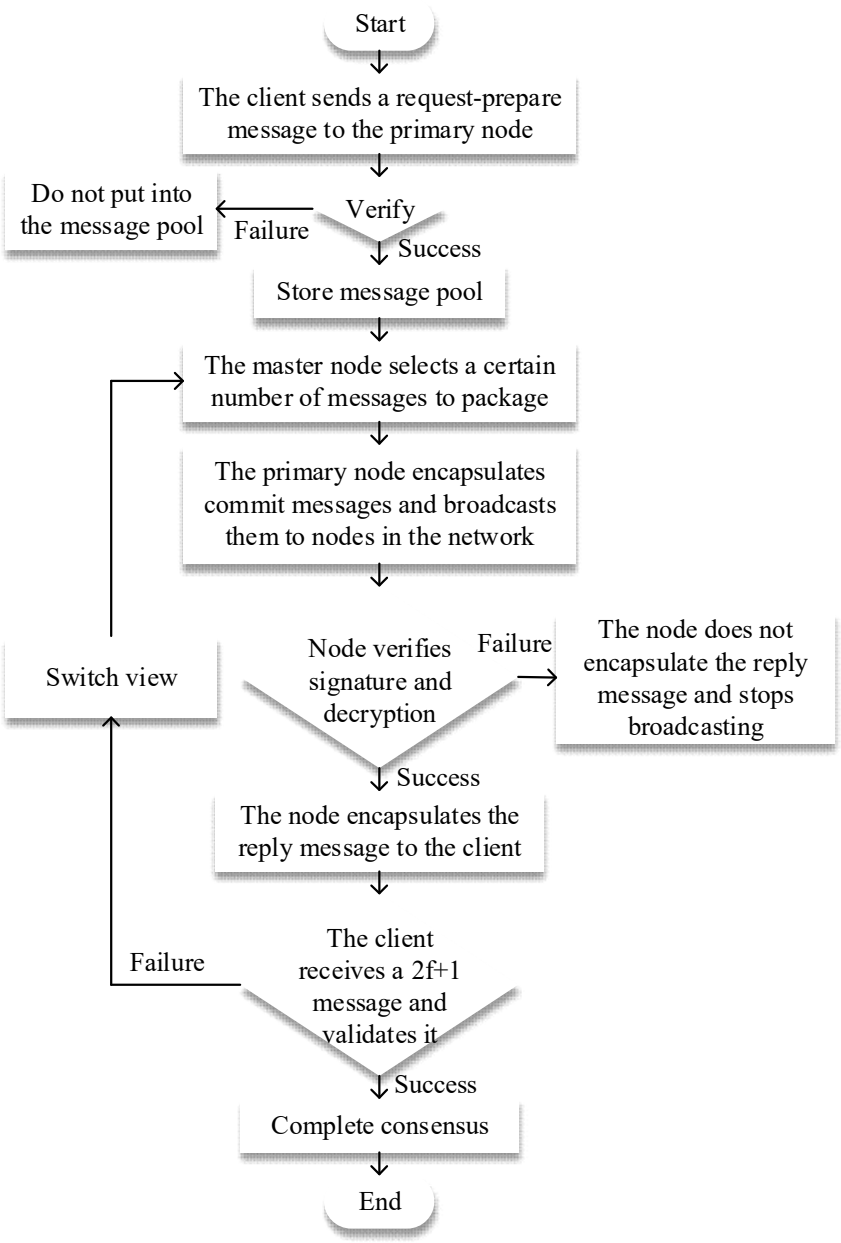

Fig. 2. RPBFT process

4. Improvement of RPBFT algorithm

4.1. The overall idea of the algorithm

Based on the PBFT algorithm, the RPBFT algorithm uses the same master node selection mechanism. Whenever a master node completes block generation and consensus for the current view, the system switches to the next view, and the master node selection rules of this mechanism are shown below.

$$p = v \bmod n \tag{10}$$

where, p is the master node number for the next round, v is the view number for this consensus round, and n is the total number of consensus nodes in the consensus network.

The R-PBFT algorithm adds record nodes, which are served by non-participating consensus nodes. Record nodes and master nodes also need to reward and punish the nodes in the network so as to update the fixed waiting time and increase the probability of reliable nodes to act as master nodes.

Secondly, in order to make the RPBFT algorithm suitable for this view switching mechanism, this paper also slightly changes the RPBFT algorithm. So that it can record the performance of the nodes in the network in real time by the relevant nodes in each round of consensus bob. At the end of the paper, experiments are conducted on the delay generated by node consensus and node campaigning for the master node, and the improved algorithm can effectively reduce the communication overhead and communication delay of the system and improve efficiency.

4.2. Improvement of the RPBFT algorithm

The R-PBFT algorithm improves the master node selection mechanism for the shortcomings of the RPBFT algorithm [22] and the PBFT algorithm for master node selection. Each node in the election of master node stage is a period of waiting time T, waiting time T contains random waiting time t and fixed waiting time t_i. the size of random waiting time t ranges from [5ms, 15ms], each node's random waiting time within this range, randomly allocated by the system, in order to prevent the phenomenon of a fixed node elected as the master node for a long time to prevent the algorithm from being centralized. For some nodes to appear downtime or malicious behavior, the system sets a fixed waiting time t_i. i of the fixed waiting time t_i represents the node number, and the upper limit of t_i is 10ms, once the fixed waiting time t_i exceeds this range, the node will be kicked out of the consensus network. And the fixed waiting time t_i size of a node is determined by the performance of that node in the consensus phase. If a node does not respond within the specified time, the fixed waiting time of that node is increased by 1ms, and thereafter the fixed waiting time of that node is increased at a rate of two times if the node has not returned to the online state. When a node sends an incorrect message, it will be determined as a malicious node by the system and the node will be expelled from the consistent network, thus ensuring the security of the whole blockchain. The specific formula is as follows.

$$T = rand(t) + t_i \tag{11}$$

In order to ensure the activity of the system, the R-PBFT algorithm improves the view switching mechanism, which is triggered when the following situations occur.

1) The system will be triggered view switching mechanism only when the master node is down or the sent message is rejected by nodes with the number at least $n/3$.

2) The system automatically triggers the view switching mechanism every 20 rounds of consensus. This ensures that each node has the possibility of being elected as the master node.

View switching has two phases, namely V-Prepare phase and V-Commit phase. Among them, the new R-node record node is added in the switching view, the record node does not participate in the consensus in the network, and is only used to record and store the fixed time and master node information of each node.

For the RPBFT algorithm along the PBFT algorithm master node switching problem, the improved R-PBFT algorithm adds a record node on top of the RPBFT, which is used to record the master node and the waiting time of each node. On this basis, the R-PBFT algorithm adds a view switching phase to ensure the vitality of the system.

4.3. Overall flow of the algorithm

The R-PBFT algorithm is improved on the basis of the RPBFT algorithm, the view switching mechanism of the RPBFT algorithm is changed, and some adaptation optimizations are also made for the consistency consensus process. The specific steps of the R-PBFT algorithm are described in detail below. The specific flow chart is shown in Figure 3.

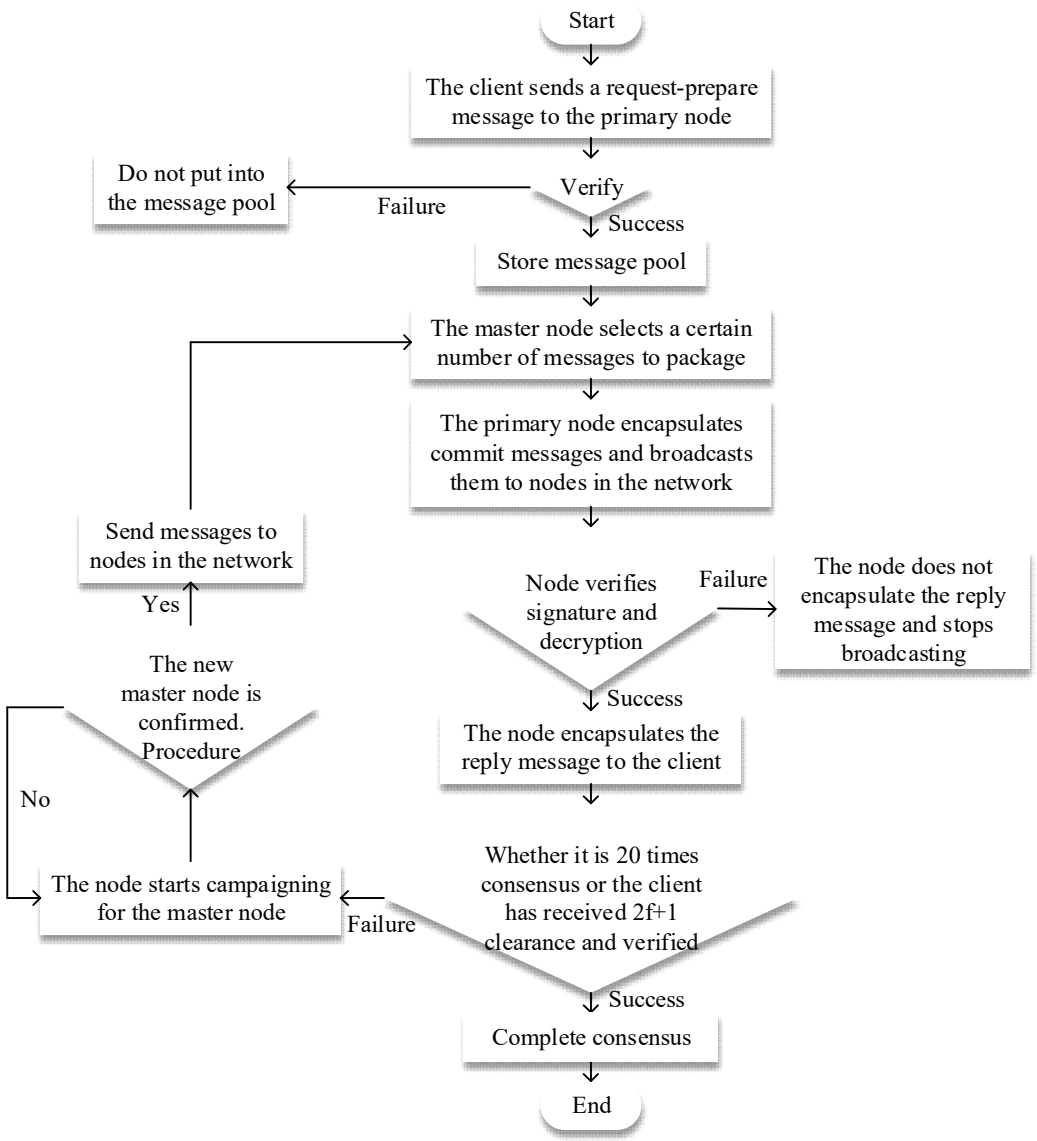

Fig. 3. R-PBFT consensus flowchart

5. Application and Experimentation of Consensus Algorithm RPbft

5.1. Application and experimental validation

The improved PBFT consensus algorithm based on RSA in this paper is applied to distributed database. This method can not only reduce the storage pressure of the database server, but also ensure that the database data can not be tampered with and the confidentiality of user data, and further optimize the consistency of the database security mechanism.

The RPbft algorithm is applied to the distributed database system, and many tests and records are made during the operation of the system to verify that the algorithm is feasible, meets the desired goals, is usable, and can meet the system requirements. After completing the blockchain logistics project using the Java programming language, different processes are used on the same machine to simulate the multi-node activities carried out in the system, and the corresponding processes are stopped when simulation operations such as node downtime are required. Due to the performance of the test machine, it led to the fact that the number of nodes in the algorithm could not grow upward indefinitely, and only some of the processes were tested as active nodes for the simulation under the existing conditions. In the experimental test, in order to ensure the accuracy of the data, and to exclude

the influence of the environment on the experimental results, after a number of experiments to take the average value of the statistics and do table plotting.

5.2. Consensus delay test

Consensus delay refers to the delay time between the client sending out a request and receiving feedback from the system that the request has been accepted successfully, the specific calculation method is T (delay) = T (number of milliseconds to receive feedback) - T (number of milliseconds to send out a request), and is limited by the influence of the hardware environment so that the consensus process, which may be very fast, has become relatively slow, but it is also just as convenient for experimenters to carry out experimental operations and data statistics. However, it is also convenient for the experimenter to conduct experimental operations and data statistics, etc. The relationship between the specific delay and the number of nodes and types of algorithms is shown in Table 1 (unit: milliseconds).

The data shown in Table 1 are the average values obtained after several experiments and removing the data affected by the environment, and the corresponding consensus delay graph is shown in Fig. 4. The consensus delay of the RPbft algorithm is basically in line with the expected expectation compared with the Raft algorithm and the Pbft algorithm when the delay increases with the number of nodes, and when the number of nodes is 52, the delay is 261 milliseconds. Since the system is tested when the leader (leader) node of the Raft algorithm did not have problems such as dropping out, the consensus delay in the Raft algorithm with the increase in the number of nodes is not very obvious, and there is only a small amount of growth trend.Pbft algorithm, on the other hand, because the consumption of communication resources by the increase in the number of nodes will grow extremely fast, so the consensus delay fluctuates more in the graph.

Table 1. Consensus schedule

Quantity	12	16	20	24	28	32	36	40	44	48	52
Pbft	210	213	217	223	231	254	261	273	316	369	410
Raft	151	150	150	152	154	155	155	157	160	164	166
RPbft	179	181	185	188	196	199	205	212	223	241	261

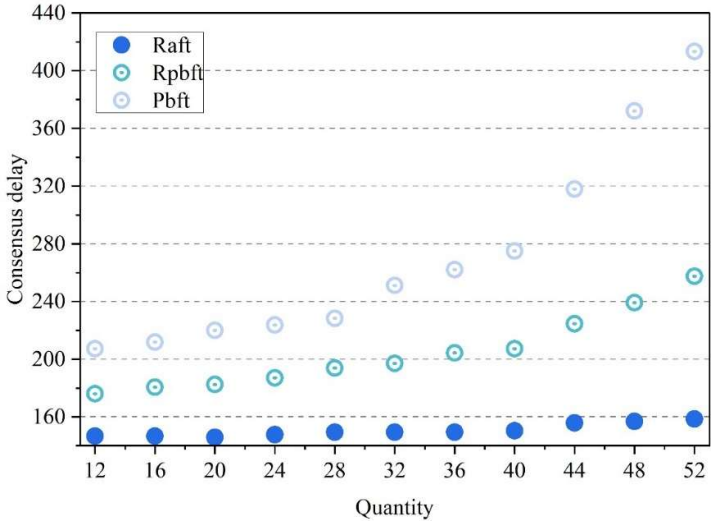

Fig. 4. Consensus delay

5.3. Recovering Consensus Latency after a Drop of the Master Node

In practical applications, no system can always be in the running state without accidents, so this system manually end the process to simulate the system running in the process of its main nodes dropped or downtime, the specific calculation method is T (delay) = T (consensus recovery) - T (end the process of the number of milliseconds), the statistical mode is the same for a number of experiments and statistics to remove data affected by the environment and take the average value. The statistical method is also to remove the average value after several experiments and statistics by removing the data affected by the environment, and the relationship between the specific delay and the number of nodes and the type of algorithm is shown in Figure 5.

After experiments and comparative analysis, it is found that the RPbft algorithm, compared with the Pbft algorithm and the Raft algorithm, is able to realize the re-operation of the system faster after the downtime of the main nodes, and when the number of nodes is 48, the delay of restoring the consensus is 364 milliseconds. The specific performance of the algorithm in the blockchain distributed database project is largely in line with expectations.

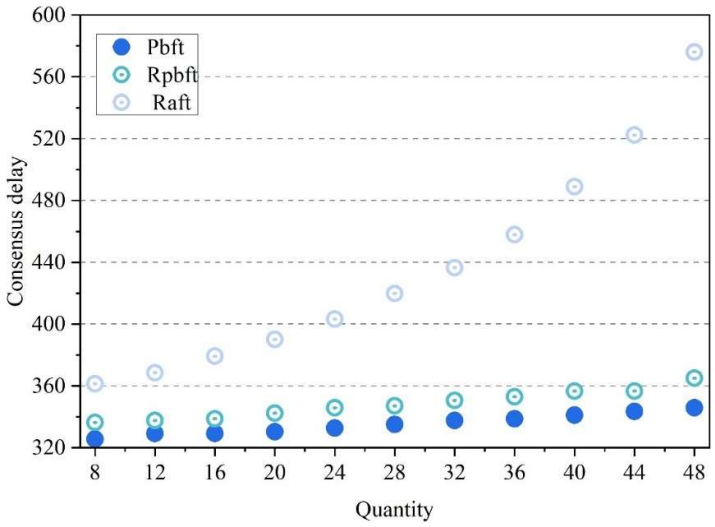

Fig. 5. The consensus is delayed

6. Performance analysis of the R-PBFT algorithm

The experimental environment is 8 hosts with 1TG hard disk, 16G RAM, and Intel i5-6500 CPU.The system used for the experiment is Ubuntu 64-bit, and the experimental platform is Hyperledger Fabric v1.0.The experiment adopts the Docker container virtualization technology, which is widely adopted in the cloud computing environment because of its lightweight and high efficiency. Docker container virtualization technology is widely used in cloud computing environments because of its lightweight and efficiency, and each Docker container represents a slice.

6.1. CPU Utilization Test

In the CPU utilization test experiment, both the software and hardware environments are kept the same, and the CPU utilization values over time are observed for the three schemes of R-PBFT algorithm, OmniLedger and RapidChain, and the results are shown in Figure 6. During the CPU utilization test of the three schemes, the OmniLedger scheme reaches 100% CPU utilization during the consensus process, which is due to the fact that each node in the system is competing for arithmetic power when consensus is performed through this scheme, resulting in high CPU utilization, and the RapidChain scheme is in the middle of the CPU utilization when consensus is performed, and the final CPU utilization rate reaches 68%. RPBFT program in the consensus CPU utilization in the three programs

in the lowest position, CPU utilization rate of only 55%, this is due to the R-PBFT algorithm program in the consensus process, the algorithm in the block transmission and signature verification efficiency has been greatly improved, the nodes do not need to compete for too much arithmetic can complete the consensus.

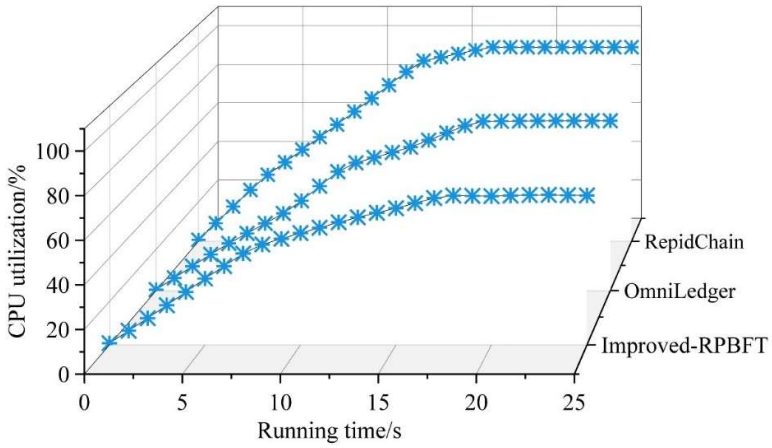

Fig. 6. Influence of Byzantine node number on block time

6.2. System Fault Tolerance Testing

In the system fault tolerance experiments, both the software and hardware environments are kept the same, and the block out time of the three schemes, R-PBFT algorithm, OmniLedger and RapidChain, is observed by continuously increasing the number of Byzantine nodes in the slice, and the results are shown in Fig. 7. When the number of Byzantine nodes in the slice increases, the blocking time of all three schemes, R-PBFT algorithm, OmniLedger and RapidChain, increases relatively, but compared with the other two schemes, the blocking time of R-PBFT algorithm scheme is always relatively low, which can be concluded that the R-PBFT algorithm scheme is better in terms of fault tolerance.

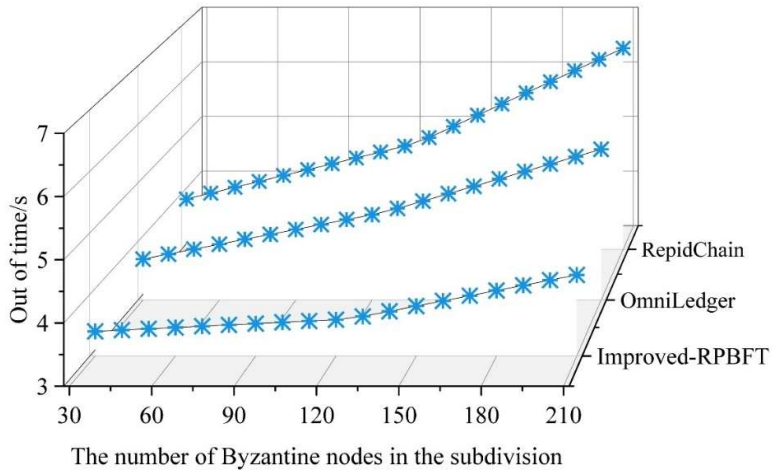

Fig. 7. Influence of Byzantine node number on block time

6.3. System Throughput Testing

In the experiment of testing the throughput of the system, by gradually increasing the proportion of Byzantine nodes in the system, the throughput of the three schemes of R-PBFT algorithm, OmniLedger and RapidChain is recorded under different proportions of Byzantine nodes, respectively, and the

hardware and software environments are kept the same in the process of testing, and the results of the test are shown in Fig. 8.

When the proportion of Byzantine nodes in the system increases, the throughput of the three schemes, R-PBFT algorithm, OmniLedger and RapidChain, decreases, but a closer look reveals that when the proportion of Byzantine nodes is small, the throughput of the OmniLedger scheme is the highest, the throughput of the RapidChain scheme is the lowest, and the RPBFT scheme is in the middle. middle, but when the proportion of Byzantine nodes continues to increase, the throughput of the OmniLedger scheme and the RapidChain scheme is rapidly decreasing, the R-PBFT algorithm scheme is also decreasing, but the throughput decreases a little bit slower, and the overall throughput is above that of the other two schemes, and the R-PBFT algorithm scheme is better, and at the time when the proportion of Byzantine nodes is 0.3, the throughput is 90/s.

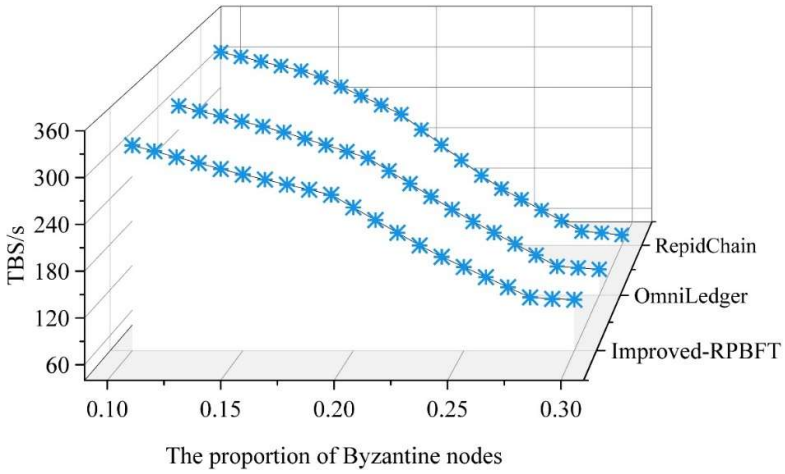

Fig. 8. Influence of different Byzantine node ratios on TPS

6.4. Transaction delay testing

In the transaction delay test, the soft and hard environments are kept the same, the number of nodes in the slice and the number of Byzantine nodes are kept the same, the number of slices in the distributed database is increased sequentially by the number of slices in the distributed database to observe the transaction delay of the three schemes of R-PBFT algorithm, OmniLedger and RapidChain, and the test results are shown in Figure 9. When the number of shards in the distributed database increases in turn, the increase in transaction latency of the three schemes of R-PBFT algorithm, OmniLedger and RapidChain, but the R-PBFT algorithm scheme has a lower transaction latency compared with the other two schemes, and the R-PBFT algorithm scheme is better in general, and when the number of shards in the distributed database system increases to 20. The transaction latency is only 32s, which is 13s and 25s lower than that of RapidChain and OmniLedger, respectively.Through the above comparison test of the three schemes of R-PBFT algorithm, OmniLedger and RapidChain, it can be seen that the scheme proposed in this paper can really solve the problem of scalability of the distributed database effectively and improve the distributed database throughput.

By comprehensively analyzing and improving the data consistency assurance mechanism in distributed database systems under cloud computing environment, this paper reveals the core challenges faced in this area and the effectiveness of its improvement strategy in this paper. Although there are certain technical challenges and trade-offs, with the continuous development and optimization of technology, a more efficient and reliable data consistency assurance mechanism will be realized in the future.

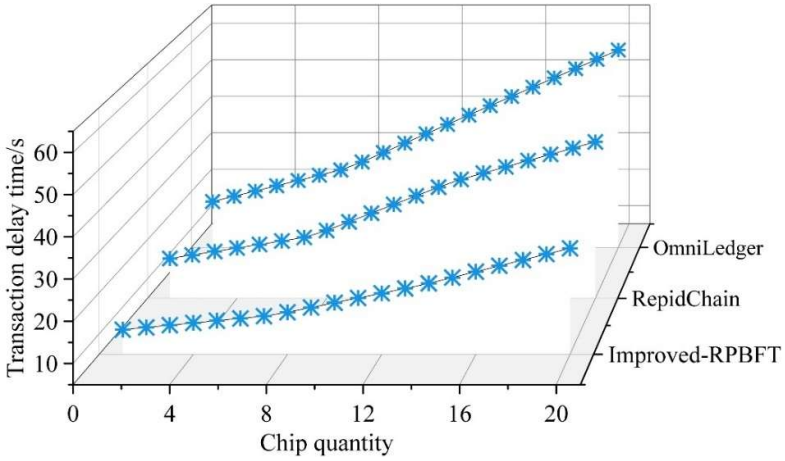

Fig. 9. Impact of number of segments on transaction delay

6.5. Scalability testing

In distributed systems, scalability is the ability of a system to maintain or increase its performance level as workloads or resources are added. Specifically, scalability means that as more nodes, more resources, or more workloads are added to a system, the performance of the system can grow linearly or nearly linearly without being affected by system bottlenecks, bottleneck nodes, or performance bottlenecks. Scalability can be categorized into horizontal scaling by increasing the number of nodes, server capacity, and cluster size and vertical scaling by increasing the system resources and physical performance of individual nodes.

The experiment is set up as a scalability test to evaluate the impact of dynamic increase or decrease of nodes during the running of the consensus algorithm by increasing the number of nodes and measuring the TPS. The client sent a 50-byte transaction at 3000 TPS, and continuously increased the number of network nodes through a timing script to calculate the TPS under the current number of nodes in 1-minute increments.This experiment tested three schemes, namely, the R-PBFT algorithm, OmniLedger, and RapidChain, and analyzed the algorithm's transaction processing performance under different numbers of nodes.

The experimental results are shown in Figure 10. With the increase of the number of nodes, OmniLedger and RapidChain have a significant performance degradation due to the communication complexity of their algorithms. When there are 1 node and 40 nodes in the system, the TPS value of OmniLedger algorithm is about 1000 and 380 respectively, i.e., the performance of 40 nodes is about three times of that of 1 node, and the performance of R-PBFT algorithm is relatively stable, and the increase in the number of nodes does not have much effect on the performance of the algorithm. The results show that OmniLedger and RapidChain algorithms are less scalable and the performance of the algorithms is greatly affected by the number of nodes, while the R-PBFT algorithm is more scalable and the performance of the algorithms is still stable in the case of multiple nodes.

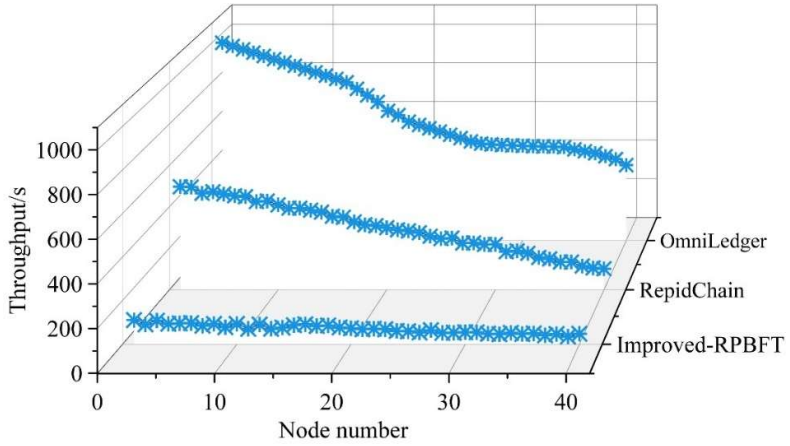

Fig. 10. Extensibility test

7. Conclusion

In this paper, strategies such as RSA encryption algorithm and improved view switching mechanism are applied to improve the PBFT algorithm, solve the defects of PBFT algorithm, and further optimize the consistency guarantee mechanism of each node under the distributed database.

1) The RPBFT algorithm is proposed to address the problems of high latency and low efficiency of the PBFT algorithm. The algorithm combines the RSA encryption algorithm and the PBFT algorithm, optimizes the consistency agreement process of the PBFT algorithm, reduces the consensus delay of the algorithm to a certain extent, and improves the consensus efficiency of the algorithm. When the number is 52, the delay of the RPBFT algorithm is 261 milliseconds, which reduces by 149s compared with that of the PBFT algorithm, and the RPBFT algorithm has a significant advantage in the practical application.

2) Subsequently, on the basis of the RPBFT algorithm, the view switching mechanism is improved and the R-PBFT algorithm is proposed. The algorithm reduces the possibility of Byzantine nodes appearing in the nodes in the consensus network, making the whole consensus process more secure and reliable. The improved view switching mechanism ensures the strong consistency of the distributed database in a highly concurrent transaction environment with low CPU utilization, and the final CPU utilization is only 55%. The algorithm also has better fault tolerance and scalability, and can reduce the consensus delay while maintaining high throughput, when the number of shards in the distributed database system is 20, the transaction delay time is lower than that of RapidChain and OmniLedger by 13s and 25s, respectively.

3) In summary, the improved algorithm in this paper shows good performance in the current experimental environment, and has significant advantages in the consistency assurance of distributed databases with blockchain technology. However, its adaptability in larger-scale networks and more complex application scenarios needs further research. With the application of blockchain's technology in more fields, such as IoT, finance, etc., the improved algorithm in this paper needs to be further optimized to adapt to different security and performance requirements.

References

[1] Fong, E. N., Sheppard, C. L., & Harvill, K. A. (2021). Distributed Database Design. In Handbook of Data Management 1999 Edition (pp. 571-581). Auerbach Publications.

[2] Qiu, X., & Wang, B. (2019, March). Analysis and Research on data transmission efficiency of distributed measurement and control system. In Journal of Physics: Conference Series (Vol. 1176, No. 4, p. 042015). IOP Publishing.

[3] Guo, Y., Wang, S., & Huang, J. (2021). A blockchain-assisted framework for secure and reliable data sharing in distributed systems. EURASIP Journal on Wireless Communications and Networking, 2021, 1-19.

[4] Petrov, A. (2019). Database Internals: A deep dive into how distributed data systems work. O'Reilly Media.

[5] Thokala, V. S. (2021). A Comparative Study of Data Integrity and Redundancy in Distributed Databases for Web Applications. Int. J. Res. Anal. Rev, 8(4), 383-389.

[6] Ivaki, N., Laranjeiro, N., & Araujo, F. (2018). A survey on reliable distributed communication. Journal of Systems and Software, 137, 713-732.

[7] Ganesan, A., Alagappan, R., Arpaci-Dusseau, A. C., & Arpaci-Dusseau, R. H. (2017). Redundancy does not imply fault tolerance: Analysis of distributed storage reactions to file-system faults. ACM Transactions on Storage (TOS), 13(3), 1-33.

[8] Zhang, D., Dai, Z. Y., Sun, X. P., Wu, X. T., Li, H., Tang, L., & He, J. H. (2024). A distributed data processing scheme based on Hadoop for synchrotron radiation experiments. Journal of Synchrotron Radiation, 31(3).

[9] Nashat, D., & Amer, A. A. (2018). A comprehensive taxonomy of fragmentation and allocation techniques in distributed database design. ACM Computing Surveys (CSUR), 51(1), 1-25.

[10] Brock, B., Cohn, R., Bakshi, S., Karna, T., Kim, J., Nowak, M., ... & Mattson, T. G. (2024, May). Distributed Ranges: A Model for Distributed Data Structures, Algorithms, and Views. In Proceedings of the 38th ACM International Conference on Supercomputing (pp. 236-246).

[11] Bieniusa, A., Gotsman, A., Kemme, B., & Shapiro, M. (2018). Data Consistency in Distributed Systems: Algorithms, Programs, and Databases. Dagstuhl Reports, 8(2), 101-121.

[12] Gomes, V. B., Kleppmann, M., Mulligan, D. P., & Beresford, A. R. (2017). Verifying strong eventual consistency in distributed systems. Proceedings of the ACM on Programming Languages, 1(OOPSLA), 1-28.

[13] Dizdarevic, J., Avdagic, Z., Orucevic, F., & Omanovic, S. (2021). Advanced consistency management of highly-distributed transactional database in a hybrid cloud environment using novel R-TBC/RTA approach. Journal of Cloud Computing, 10, 1-31.

[14] Krechowicz, A., Deniziak, S., & Łukawski, G. (2021). Highly scalable distributed architecture for NoSQL datastore supporting strong consistency. IEEE Access, 9, 69027-69043.

[15] Abbaszadeh, M., & Saeedvand, S. (2018). Weak Consistency Model in Distributed Systems Using Hierarchical Colored Petri Net. J. Comput., 13(2), 236-243.

[16] Aikins, M. V. (2023). Distributed storage systems and how they handle data consistency and reliability. Faculty of Natural and Applied Sciences Journal of Scientific Innovations, 5(1), 83-89.

[17] Shukur, H., Zeebaree, S., Zebari, R., Ahmed, O., Haji, L., & Abdulqader, D. (2020). Cache coherence protocols in distributed systems. Journal of Applied Science and Technology Trends, 1(2), 92-97.

[18] Ramin Sadooghi,Taher Niknam,Morteza Sheikh,Jamshid Aghaei,Vahid Vahidinasab,Om Malik & Saeed Fotovat. (2025). Peer-to-peer energy management of distributed ledgers in renewable smart energy systems. Electric Power Systems Research111451-111451.

[19] Mohammad Saidur Rahman, Ibrahim Khalil, Mohammed Atiquzzaman & Abdelaziz Bouras. (2025). A lightweight practical consensus mechanism for supply chain blockchain. High-Confidence Computing(1), 100253-100253.

[20] Zhe Li,Jinsong Wang & Yi Li. (2024). An improved PBFT consensus algorithm based on reputation and gaming. The Journal of Supercomputing(1),323-323.

[21] Wan Nur Aqlili Ruzai,Amir Hamzah Abd Ghafar,Muhammad Rezal Kamel Ariffin & Muhammad Asyraf Asbullah. (2025). On Newly Partial Key Exposure Attack Using Near-square RSA Primes. IAENG International Journal of Applied Mathematics(2),

[22] Luo Haoxiang,Yu Hongfang & Luo Jian. (2023). PRAFT and RPBFT: A class of blockchain consensus algorithm and their applications in electric vehicles charging scenarios for V2G networks. Internet of Things and Cyber-Physical Systems61-70.