Raft-S: A Sharding Consensus Algorithm Based on Queue Theory and the Raft

Yiqin Chen[1], Ruowen Gu[1], Dongyan Huang[1(✉)], and Yong Ding[2]

[1] Ministry of Education Key Lab. of Cognitive Radio and Information Processing, Guilin University of Electronic Technology, Guilin 541004, Guangxi, China
Huangdongyan-gua@163.com

[2] Guangxi Key Laboratory of Cryptography and Information Security, Guilin 541004, Guangxi, China

Abstract. The Raft is a strong leader consensus algorithm, where the leader is responsible for consensus decisions such as log replication, so the leader is under greater pressure to become a performance bottleneck of the Raft, leading to a limited scale and insufficient scalability of the Raft. Firstly, a sharding consensus algorithm Raft-S is proposed, which linearly increases the scalability and TPS (Transactions per second) ensuring global consistency and security through 2PC (Two-phase commit) protocol and monitoring nodes. Secondly, the PDF (Probability Distribution Function) of consensus time of each shard is modeled using the followers' delay distribution, thereby measuring the consensus efficiency of shards. Finally, a real-time transaction distribution strategy based on queuing theory is proposed to avoid the transaction imbalance.

Keywords: Raft · Queue Theory · sharding

1 Introduction

Blockchain is a technology that uses cryptography and consensus algorithms to establish a global trust relationship in a P2P (peer-to-peer) network. It has broad application prospects, such as financial supervision, data sharing, and government auditing. The consensus algorithm is the protocol to be executed to reach consensus in the distributed system, which directly determines the service quality of the blockchain. However, existing consensus algorithms have certain performance bottlenecks such as low efficiency and scalability, the blockchain has not been implemented on a large scale at present.

Supported by the Guangxi NSF Project under Grant 2022GXNSFBA035645; by the Guangxi Key Research and Development Program under Grant AB20238026; by the Chinese NSF Project under Grant 6217070229; by the Director Fund of Key Laboratory of Cognitive Radio and Information Processing, Ministry of Education (Guilin University of Electronic Technology) under Grant CRKL210104; and by the Innovation Project of GUET Graduate Education under Grant 2022YCXS052.

J. Wang (Ed.): MobiMedia 2023, LNICST 576, pp. 357–373, 2025.
https://doi.org/10.1007/978-3-031-60347-1_30

Raft is a distributed consensus algorithm proposed by Diego Ongaro in 2014 [1], by setting up a leader responsible for consensus decisions such as log synchronization to reach consistency. Because of its simple architecture and high efficiency, Raft has become the mainstream algorithm of the consortium blockchain, such as Quorum [2]. However, Raft is a strong leader algorithm, the leader is under greater pressure than the followers in communication and load. The leader becomes a performance bottleneck, resulting in a limited Raft scale in practical deployment [3], so Raft is more suitable for small private blockchains. Raft's consensus efficiency decreases as the system size or fault nodes increase, so the scalability and fault tolerance of Raft needs to be improved to suit IoT (Internet of Things).

Optimizing Raft's performances is a hot topic of current research. For example, Ermin *et al.* [4] used SDN (Software Defined Network) to improve the consistent response time of the Raft algorithm. Tian *et al.* [5] proposed a Byzantine fault-tolerant algorithm B-Raft using the Schnorr signature mechanism. Huang *et al.* [6] proposed RBFT with high TPS and Byzantine fault tolerance based on network sharding. Gu *et al.* [7] proposed a leadership transfer algorithm to avoid network splitting in Raft. However, there is still a lack of concern about Raft's scalability.

The sharding technology has become an important solution to improve the scalability of the blockchain through the on-chain horizontal expansion. It comes from the database field, by dividing large databases into multiple small partitions for easy data management, such as Megastore [8], Spanner [9], and Scanner [10]. In the blockchain, the sharding technology breaks the performance bottleneck of a single group by delivering different transactions to multiple groups for parallel consensus, as shown in Fig. 1.

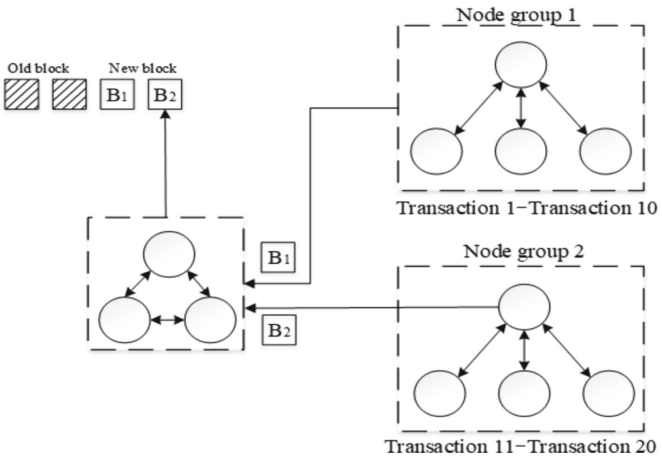

Fig. 1. The sharding technology in blockchain.

Zheng et al. [11] proposed a sharding consortium blockchain Meepo to improve the security and effectiveness of the sharding technology. To reduce the aggregation of malicious nodes, Xu et al. [12] proposed a sharding selection algorithm based on the evolutionary game. Bai et al. [13] proposed a hierarchical blockchain architecture and

designed a blockchain sharding algorithm, which improved the TPS of the blockchain and shortened the sharding consensus time. Wang et al. [14] proposed a reputation-based consensus protocol for blockchain sharding, which prevents malicious nodes from concentrating on a single shard. Yoo et al. [15] divided the blockchain into multiple shards based on domain and makes the blockchain more credible by changing the committee members. Kim et al. [16] used the two-phase cooperative bargaining game model to achieve maximum efficiency for the blockchain using sharding technology.

The sharding technology can linearly improve Raft's scalability, and Raft does not need high-cost methods to reach consensus such as mining, so the consensus environment is more secure. Once consensus is completed within a shard, the shard only needs to broadcast the consensus result in time to ensure global consistency and security. Therefore, we propose a sharding consensus algorithm Raft-S by combining sharding technology with Raft, which linearly improves Raft's capacity. And a transaction distribution strategy is proposed to avoid transaction imbalance based on queuing theory. The main contributions are as follows:

1. The sharding consensus algorithm Raft-S with monitoring nodes and distribution node is proposed, and its security and consistency are analyzed.
2. Build a queuing theory model for the Raft-S, transactions are regularly distributed to each shard to avoid the transactions imbalance, thus achieving the maximum consensus efficiency.
3. A method for calculating PDF of sharding consensus time according to the follower's delay distribution.

The rest are organized as follows. Section 2 introduces the Raft-S consensus algorithm in detail. Section 3 proposes the method for calculating the PDF of the sharding consensus time, and a queuing theory model suitable for the Raft-S is described. Section 4 obtains the PDF of sharding consensus time and analyses the transaction distribution strategy. Section 5 gives conclusions.

2 Raft-S Consensus Algorithm

As the system scale increases, the Raft efficiency decreases, which indicates that Raft has insufficient scalability. The sharding technology divides the consensus nodes into non-overlapping shards, which execute the consistency protocol in parallel to realize the linear increase in the capacity or TPS. Network sharding means that transactions are randomly allocated to different shards, which is prone to create transaction imbalance when sharding consensus efficiency is different. For example, the shard with high consensus efficiency is idle, and the shard with low efficiency is blocked, thus affecting the consensus efficiency.

We have designed a sharding consensus algorithm Raft-S with a distribution node. The distribution node allocates transactions for each shard according to the queuing theory model applicable to Raft-S, which solves the transaction imbalance and achieves maximum efficiency. In addition, to address the excessive power of leader we have introduced anonymous monitoring nodes in the shards to prevent leaders from malicious tampering with transactions.

2.1 Raft-S Consensus Algorithm

If the total number of nodes is $N + 1$, one distribution node, and the number of shards is k, then the number of nodes in each shard is N/k, including one leader, 2 monitoring nodes, and (N/k)-3 followers. Raft-S consensus algorithm is as follows (Fig. 2):

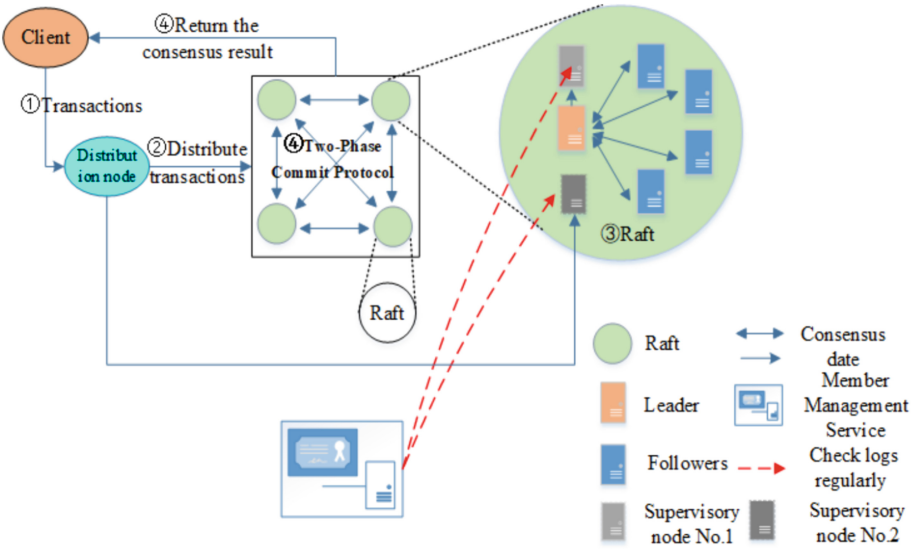

Fig. 2. Raft-S Consensus Algorithm.

The client sends transactions to the distribution node, and the node distributes the transactions to individual shards and corresponding supervisory nodes according to transactions distribution strategy. After receiving the transaction, the leader puts the transaction detail into RPC (Remote Procedure Call) and sends it to the follower [1]. After reaching consensus, the shard followers apply the entry to their own state machine. The shard leader will return the consensus results to the client and broadcast the changes in the state machine to other shard leaders for global consistency.

2.2 Global Consistency and Security of Raft-S

2PC protocol ensures the strong consistency of distributed systems by dividing the processing of transactions into two phases: voting and committing [17]. In the traditional 2PC protocol, only one server is the coordinator, and the other servers are participants, as shown in Fig. 3. Because of the parallel processing of transactions in Raft-S, we updated this protocol. All shard leaders are both coordinators and participants, as shown in Fig. 4.

After the transactions have reached consensus within the shard, the shard leader will perform both of the following operations: 1. Return the consensus result (e.g., OK) to the client; 2. Changes in the state machine are broadcast to other shard leaders via the

2PC protocol. After receiving the broadcast, the other shard leaders send the changes to their followers to achieve global consistency. It is important to note that the other shards do not need to consensus on the changes again, but simply read and apply it to the state machine.

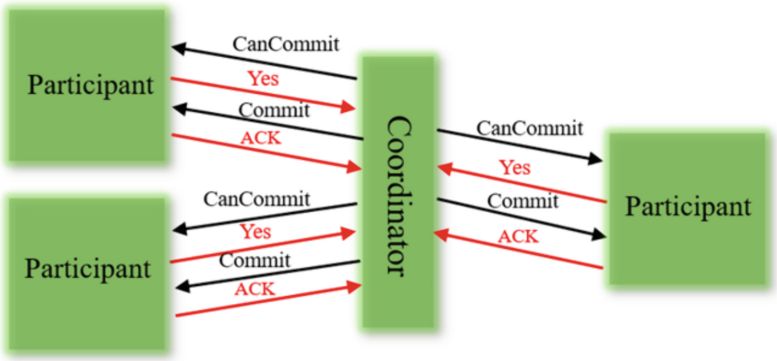

Fig. 3. Traditional 2PC protocol.

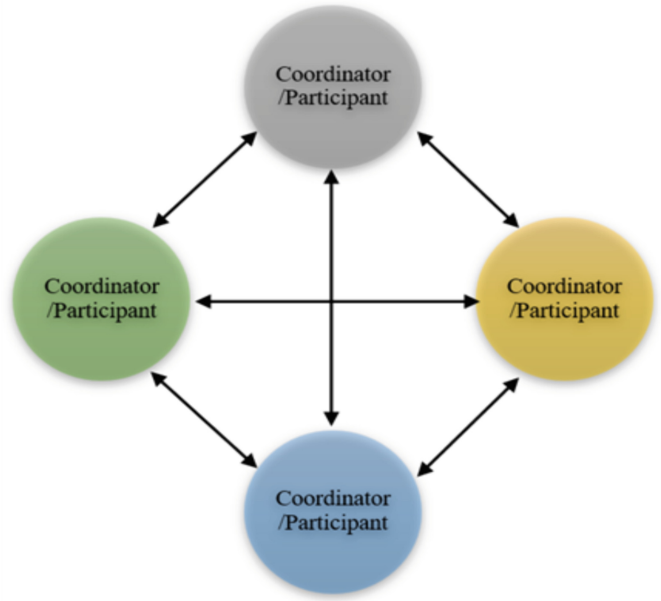

Fig. 4. 2PC protocol in Raft-S.

2.3 Security of Raft-S

After In the Raft-S, there are two anonymous monitoring nodes inside each shard that do not participate in consensus and have no voting rights. Monitoring node No.1 receives logs from leaders. Monitoring node No. 2 receives the same transaction from the distribution node and adds the transaction details to its own logs. The monitoring nodes do not apply the logs to the state machine or clear the expired logs but act as a database to facilitate member management service to query the logs.

The member management service periodically queries the added logs and compares the logs to judge whether the shard leader has maliciously tampered with the transactions. If the logs of the two monitoring nodes are inconsistent, the shard leader has maliciously tampered with the transactions. The member management service kicks the malicious leaders out of the cluster, and the shard performs the leader election to elect a leader.

Raft-S can linearly increase the system capacity and TPS of Raft, so that its application scenarios are no longer limited by scalability and fault tolerance. And with the introduction of distribution node and monitoring nodes, Raft-S avoids transactions imbalance and decentralizes the leader's power, providing greater security.

3 Queueing Theory Model Applicable to Raft-S

3.1 Service Rate of Shards

There are s followers in a shard, and $((N/k) - 3)/2 \leq s \leq (N/k) - 3$. When the shard leader receives the reply from at least $((N/k) - 3)/2$ followers, it represents that the shard completes consensus on a transaction. Assume that the two-way communication time (later referred to as follower's delay) between the s followers and the shard leader obey the Poisson distribution with parameters $[t_1, t_2, \ldots, t_s]$, and $t_1 < t_2 < \ldots < t_s$. According to the Raft consensus property, we obtained the PDF of sharding consensus time by using the followers' delay distribution. The specific derivation process is as follows:

① In a shard, the cumulative probability $P(T \leq t)$ refers to the probability that a follower has replied to the leader before the t-th second (the probability or event referred to below is before the t-th second if not otherwise stated). Then $P(T \leq t \mid ((N/k)\text{-}3)/2 \leq i \leq s)$ is the probability that at least $((N/k) - 3)/2$ followers have replied to the leader. According to the probability relationship of complementary events, $P(T \leq t \mid 0 \leq j < ((N/k)\text{-}3)/2)$ is the probability that the leader does not receive $((N/k) - 3)/2$ followers' replies, which indicates that the shard has not completed consensus.

② Numbering the followers from 1 to s makes the modeling more organized, but it's not needed in code. The replying probability of s followers are $p_1^t, p_2^t, \ldots, p_s^t$.

③ The event with the shard has j $(0 \leq j < ((N/k) - 3)/2))$ replies of followers is called A_j. The shard does not complete consensus when A_j occurs, $A_j = A_0 \cup A_1 \cup A_2 \cup \ldots \cup A_{(((N/k)-3)/2)-1}$.

④ A_0 means there is no reply from the followers. A_1 means that only one follower replies to the shard leader. On the premise that A_1 occurs, the event that follower No.1 replies is called A_{11}, and so on, $A_1 = A_{11} \cup A_{12} \cup \ldots \cup A_{1m} \ldots \cup A_{1s}$, so we

can obtain the A_1's probability,

$$\mathrm{P}(A_1) = \sum_{m=1}^{C_s^1} P(A_{1m}). \tag{1}$$

⑤ A2 means that there are two followers who reply to the shard leader. The followers have different replying probabilities, it is necessary to calculate the A_2's probability by permutating and combining the followers. We defined the order of elements and events.

Definition 1. The order of elements: mapping the cumulative replying probability of the No.1-No.s followers set as $1 \times s$ matrix and giving order to the matrix elements, the matrix is $\left[p_1^t, p_2^t, \ldots\ldots, p_m^t\right]$.

Definition 2. The order of events: when there are j $(0 \leq j < ((N/k) - 3)/2)$ followers replying, we must determine which j followers have replied. According to the combination equation, there are C_s^j combination methods and we called the combining event is A_{jm}, $m \in [1, C_s^j]$. Starting with the first element, there is an order between the events obtained by traversing and combining elements from front to back.

For example, in the A_2, the event in which followers No. 1 and No. 2 is A_{21}, the event in which followers No. 1 and No. 3 is A_{22}, and so on, the event in which followers No. $(s-1)$ and No. s is $A_{2C_s^2}$, then the A_2's probability is

$$\mathrm{P}(A_2) = \sum_{m=1}^{C_s^2} P(A_{2m}). \tag{2}$$

⑥ The $\mathrm{A}_{(((N/k)-3)/2)-1}$'s probability is

$$\mathrm{P}\left(A_{\frac{\frac{N}{k}-3}{2}-1}\right) = \sum_{m=1}^{C_s^{\frac{\frac{N}{k}-3}{2}-1}} P\left(A_{(\frac{\frac{N}{k}-3}{2}-1)m}\right). \tag{3}$$

⑦ The probability P_N that the shard does not complete consensus is equal to the A_j's probability,

$$P_N = \sum_{m=1}^{C_s^j} \sum_{j=1}^{\frac{\frac{N}{k}-3}{2}-1} P(A_{jm}). \tag{4}$$

⑧ Therefore, the probability of the shard completing consensus before the t-th second is P_Y,

$$P_Y = 1 - P_N = 1 - \sum_{m=1}^{C_s^j} \sum_{j=1}^{\frac{\frac{N}{k}-3}{2}-1} P\left(A_{jm}\right). \tag{5}$$

If the probability of the shard completing consensus before the (t-1)-th second is $P_{Y\text{-}1}$, then the probability of the shard completing consensus at t-th second is $P_t = P_Y - P_{Y\text{-}1}$.

Through the above equations, we can obtain the probability distribution of the sharding consensus time. This allows us to distribute a suitable number of transactions for each shard to achieve higher consensus efficiency.

3.2 Transaction Arrival Rates of Shards

The process that each shard processing transactions is a Birth-Death process. The relationship between service rate and the transaction arrival rate of the shard determines the congestion degree in the queuing system. According to queuing congestion control theory, when the ratio of transaction arrival rate to service rate is equal to 1/2, the system goes from no queuing at all to queuing. When the ratio is 1, the queue tends to infinity and the queuing system is completely blocked [18]. As shown in Fig. 5.

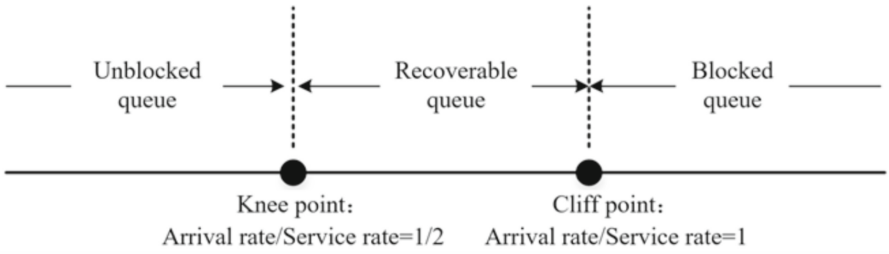

Fig. 5. Knee Point and Cliff Point in Queue Congestion Control Theory.

In summary, we assume that the distribution node determines the transaction arrival rate μ_i of each shard according to their service rate t_i. And μ_i takes a value slightly greater than $t_i/2$.

3.3 Queueing Theory Model

The consensus efficiency of each shard is different. If the same number of transactions are distributed for each shard or randomly, it is easy to cause transactions imbalance, and the system cannot achieve the ideal efficiency. We construct a single-queue multi-server parallel queuing model suitable for Raft-S. Treating transactions as customers and shards as servers, transactions are allocated in real-time by calculating the optimal queue length of the shards and the social revenue.

Overall Queuing Model

The transactions reach the distribution node with the overall transaction arrival rate, and the distribution node determines transaction arrival rate of the shards according to their service rate, as shown in Fig. 6.

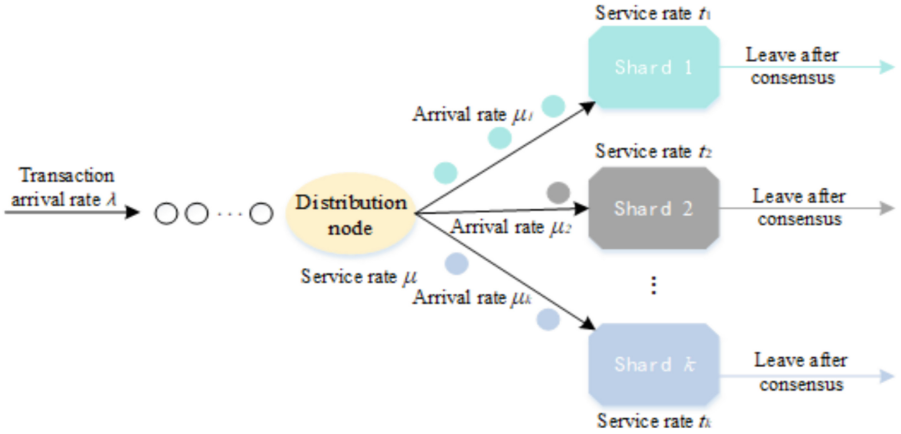

Fig. 6. Queuing theory model for Raft-S.

We establish a two-stage queue model based on the changes in transaction arrival rates. In the first stage, the transactions arrive at the distribution node and wait to be allocated, and the service rate is the number of transactions forwarded by the distribution node within one second.

The second stage is the process of reaching consensus on transactions after the transactions are allocated to the shards. We introduced the social revenue to represent the consensus efficiency of the shards. Each transaction obtains the revenue R after completing consensus, and the unit waiting cost is C. Under this mechanism, when the transactions queue of the shard reaches a certain length, the social revenue of the shard is negative. Therefore, we calculate the optimal queue length of each shard to obtain the max social revenue.

Assuming the number of shards is 4, the second stage is modeled as an M/M/4/N_i queue model ($i = 1, 2, 3, 4$), in which transaction arrival rates obey parameters μ_i, and service rates obey parameters t_i. Before the detailed description of the queuing theory model, we give the necessary symbols and descriptions (Table 1).

M/M/1 Model

Assuming that the transaction arrival rate in the first stage obeys a Poisson distribution with parameter λ and the service rate obeys a Poisson distribution with parameter μ, the state transition process of the first stage is shown in Fig. 7.

Where j represents that there are j transactions in the system, and the M/M/1 model equilibrium equation can be obtained from the state transition process,

$$\begin{cases} \pi_0\lambda = \pi_1\mu \\ \pi_0\lambda + \pi_2\mu = (\lambda + \mu)\pi_1 \\ \pi_{j-1}\lambda + \pi_{j+1}\mu = (\lambda + \mu)\pi_j, j \geq 1 \\ \sum_{j=0}^{\infty} \pi_j = 1 \end{cases}. \tag{6}$$

Table 1. Queuing Model Symbol Descriptions

Symbol	Description	Symbol	Description
λ	Overall transaction arrival rate	ρ_i	Service intensity of the *i*-th shard
μ	Service rate of the distribution node	N_i	The optimal queue length of the i-th shard
μ_i	Transaction arrival rate of the *i*-th shard	C	Unit time waiting cost
t_i	Service rate of the *i*-th shard	R	Rewards for reaching consensus
λ_{ei}	Effective transaction arrival rate of the *i*-th shard	$S(n)$	Average social revenue per unit time
ρ	Service intensity of distribution node		

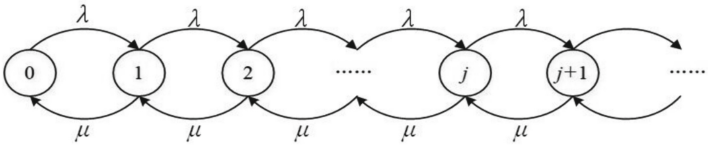

Fig. 7. State transition process of M/M/1 model.

where π_j is the probability that there are j transactions in the system. From Eq. (6), π_j can be obtained when a transaction arrives the system,

$$\begin{cases} \pi_0 = 1 - \rho \\ \pi_j = (1-\rho)\rho^j, j \geq 1 \end{cases}. \tag{7}$$

where $\rho = \lambda/\mu$ denotes the first stage service intensity, and the system is stable when $\rho < 1$. The average queue length of the first stage $E(L_{s1})$ can be calculated,

$$E(L_{s1}) = \sum_{j=0}^{\infty} j\pi_j = \rho\pi_0 \frac{d(\sum_{j=1}^{n} \rho^j)}{d\rho} = \frac{\rho}{1-\rho}. \tag{8}$$

When the system is stable, Average queue length is the number of transactions in the system that are queued up (waiting for processing),

$$E(L_{q1}) = \sum_{j=1}^{\infty} (j-1)\pi_j = \rho\pi_0 \frac{d\left(\sum_{j=1}^{n} \rho^j\right)}{d\rho} - \pi_0 \sum_{j=1}^{\infty} \rho^j = \frac{\rho^2}{1-\rho}. \tag{9}$$

According to Little's law [19], we can obtain the average duration of transactions in the system,

$$E(W_{s1}) = \frac{E(L_{s1})}{\lambda} = \frac{\frac{\rho}{1-\rho}}{\lambda} = \frac{1}{\mu - \lambda}. \tag{10}$$

At the same time, the average time spent by transactions in the queue is the average duration of transactions minus the time for transactions is served, it can be expressed as

$$E\left(W_{q1}\right) = E(W_{s1}) - E(s) = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\rho}{\mu - \lambda}. \tag{11}$$

M/M/k/N_i Model

In the second stage of Raft-S, the transactions are forwarded to each shard by the distribution node, and Raft consensus protocol is executed in each shard. When there are N_i transactions in the shard, the distribution node no longer forwards the transactions to the shard, then the equilibrium equation in the i-th shard is as follows,

$$\begin{cases} \pi_0 \mu_i = \pi_1 t_i \\ \pi_{j-1}\mu_i + \pi_{j+1} t_i = (\mu_i + t_i)\pi_j, 1 < j < N_i - 1 \\ \pi_{N_i - 1}\mu_i = \pi_{N_i} t_i \\ \sum_{j=0}^{N_i} \pi_j = 1 \end{cases}. \tag{12}$$

The probability that 0 or N_i transactions exist in the shard is

$$\begin{cases} \pi_0 = \frac{1-\rho_i}{1-\rho_i^{N_i+1}} \\ \pi_{N_i} = \frac{(1-\rho_i)\rho_i^{N_i}}{1-\rho_i^{N_i+1}} \end{cases}. \tag{13}$$

When there are N_i transactions in the shard, no new transactions are allocated to the shard, then its effective transactions arrival rate is $\lambda_{ei} = \mu_i (1 - \pi_{N_i})$. The average queue length of the i-th shard can be expressed as

$$E(L_{si}) = \sum_{j=0}^{N_i} j\pi_j = \rho_i \pi_0 \frac{d(\sum_{j=1}^{N_i} \rho_i^j)}{d\rho_i} \tag{14}$$

$$= \rho_i \pi_0 \left[\frac{\left(1 - \rho_i^{N_i+1}\right) - (N_i + 1)(1 - \rho_i)\rho_i^{N_i}}{(1 - \rho_i)^2} \right] \tag{15}$$

$$= \frac{\rho_i}{1 - \rho_i} - \frac{(N_i + 1)\rho_i^{N_i+1}}{1 - \rho_i^{N_i+1}}. \tag{16}$$

Currently, the average social revenue $S(n_i)$ per unit time of the i-th shard is

$$S(n_i) = \mu_i(1 - \pi_n)R - CE(L_{s2}) \tag{17}$$

$$= \frac{(1 - \rho_i^{N_i})\mu_i}{1 - \rho_i^{N_i+1}} R - C\left[\frac{\rho_i}{1 - \rho_i} - \frac{(N_i + 1)\rho_i^{N_i+1}}{1 - \rho_i^{N_i+1}} \right]. \tag{18}$$

To maximize $S(n_i)$, it is necessary to make

$$\begin{cases} S(n_i) \geq S(n_i - 1) \\ S(n_i) > S(n_i + 1) \end{cases}. \tag{19}$$

Simplified Eq. (19):

$$\begin{cases} \left[(\rho_i + {\rho_i}^{-1} + 2)\dfrac{R\mu_i}{C} + \rho_i\right] \geq N_i(1-\rho_i) + {\rho_i}^{N_i+1} \\ \left[(2 - \rho_i - {\rho_i}^{-1})\dfrac{R\mu_i}{C} + 1 - 2\rho_i\right] < N_i(\rho_i - 1) + {\rho_i}^{N_i+2} \end{cases} \tag{20}$$

The Eq. (20) can be regarded as ${a_i}^x + b_i x = c_i$, in which $x = N_i$, $a_i = \rho_i$, $b_i = \begin{cases} {\rho_i}^{-1} - 1 \\ {\rho_i}^{-1} - {\rho_i}^{-2} \end{cases}$, $c_i = \begin{cases} (1 + {\rho_i}^{-2} + 2{\rho_i}^{-1})\frac{R\mu_i}{C} + 1 \\ (2{\rho_i}^{-2} - {\rho_i}^{-1} - {\rho_i}^{-3})\frac{R\mu_i}{C} + {\rho_i}^{-2} - 2{\rho_i}^{-1} \end{cases}$. Using Lambert W Function [20], when $p > 0$ and $a, c \neq 0$, the equation $p^{ax+b} = cx + d$ generates the following solution,

$$x = -\frac{W\left(-\frac{lnp}{c}p^{b-\frac{ad}{c}}\right)}{alnp} - \frac{d}{c}. \tag{21}$$

Making $p \to a, a \to 1, b \to 0, c \to -b, d \to c$,

$$x = \frac{c}{b} - \frac{W\left(\frac{lna}{b}a^{\frac{c}{b}}\right)}{lna}. \tag{22}$$

Combining the Eq. (20), we can get the range of N_i,

$$\begin{cases} \dfrac{(1+{\rho_i}^{-2}+2{\rho_i}^{-1})\frac{R\mu_i}{C}+1}{{\rho_i}^{-1}-1} - \dfrac{W\left(\frac{ln\rho_i}{{\rho_i}^{-1}-1}\rho_i^{\frac{(1+{\rho_i}^{-2}+2{\rho_i}^{-1})\frac{R\mu_i}{C}+1}{{\rho_i}^{-1}-1}}\right)}{ln\rho} \geq n \\ \dfrac{(2{\rho_i}^{-2}-{\rho_i}^{-1}-{\rho_i}^{-3})\frac{R\mu_i}{C}+{\rho_i}^{-2}-2{\rho_i}^{-1}}{{\rho_i}^{-1}-{\rho_i}^{-2}} - \dfrac{W\left(\frac{ln\rho_i}{{\rho_i}^{-1}-{\rho_i}^{-2}}\rho_i^{\frac{(2{\rho_i}^{-2}-{\rho_i}^{-1}-{\rho_i}^{-3})\frac{R\mu_i}{C}+{\rho_i}^{-2}-2{\rho_i}^{-1}}{{\rho_i}^{-1}-{\rho_i}^{-2}}}\right)}{ln\,\rho_i} < n \end{cases} \tag{23}$$

The N_i can be obtained by rounding n.

4 Simulation and Analysis

4.1 Consensus Efficiency of Raft Sharding Predicting the Nodes Failure Time

Parameter and Experimental Environment

Suppose the shard size is 9, including 1 leader, 2 monitoring nodes and 6 followers. When the leader receives replies from at least 3 followers, the shard consensus is completed. The 6 follower delays obey the Poisson distribution with parameters [20, 25, 30, 40, 45, 50] respectively, and set the range of sharding consensus time to 0–80 ms according to the CDF (Cumulative Distribution Function) of the followers' delay.

The calculation process runs on a core i7, 8-core, 3.00 GHz, 48 GB memory server, and the development environment is Matlab2018.

Fitting Process

It can be seen from the Fig. 8 that the probability of each follower to reply to the leader before the 70 ms has basically reached 1. Mapping the CDF of each follower to the matrix and executing the sharding consensus time model code to obtain the CDF and PDF of sharding consensus time. The pseudo code is shown in Table 2.

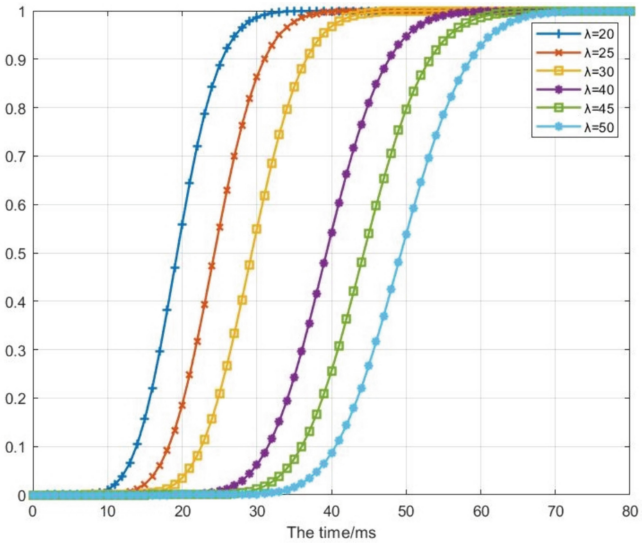

Fig. 8. The CDF of each follower replying to the leader.

Figures 9 and 10 is the CDF and PDF of sharding consensus time, the abscissa is the time. From Fig. 9, we can see that the probability of the shard completing consensus before the 50 ms reaches 1. And the PDF of the sharding consensus time roughly obeys to the Gaussian distribution from Fig. 10. We found the best parameters through fitting tool.

Using the Gaussian distribution to fit the PDF of the sharding consensus time, and the Gaussian equation is as follows,

$$y = a \cdot exp(-(\frac{x-b}{c})^2). \tag{24}$$

The fitting results show that a = 0.09525, b = 30.42, c = 5.945. The sharding consensus time obeys the Gaussian distribution with parameters (30.42, 2.9725). That is, the average time for this sharding to reach consensus on a transaction is 30.42ms.

4.2 Transactions Distribution Strategy and TPS of Raft-S

Transactions Distribution Strategy

In the common single-queue multi-server model, customers can see the status of the

Table 2. Pseudo code of sharding consensus time model

Input: The number of followers and parameters Output: the CDF and PDF of sharding consensus time
1: Set NodeNum and λ_i, $i \in$ [0, NodeNum] //Set the number of followers and their parameters.
2: $t = 0 : T$ // Define time range.
3: P_i = Poisscdf(λ_i) //Obtain the CDF of each follower.
4: for t =0 : T
5: Reply=[P_i]//Mapping the CDF of each follower into the matrix.
6: NoReply=1-[P_i]
7: for n=1: (NodeNum/2)-1 //The probability that only n followers has replied before the t ms is obtained through matrix operations.
8: NodeRe=prod(combntns(Reply, n),2)
9: NodeNoRe=prod(rot90(combntns(NoReply, NodeNum-n), -2), 2)
10: Only(n)=dot(NodeRe, NodeNoRe)
11: end
12: NoCon=sum(Only, 2);
13: Con=1-NoCon; //The CDF of sharding consensus time.
14: Con (x)=Con (x+1) - Con (x)//The PDF of sharding consensus time.
15: end

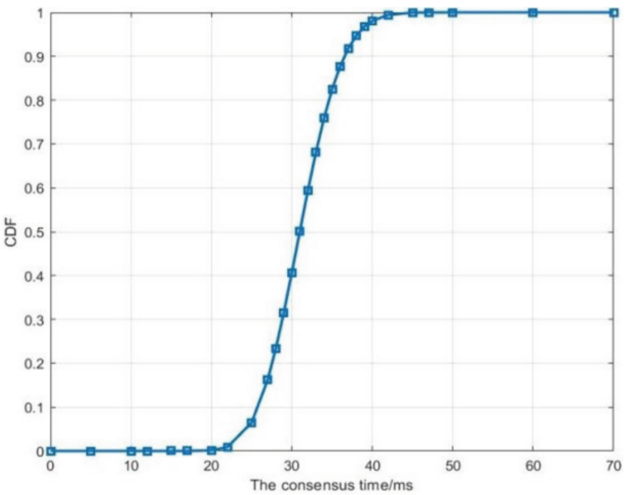

Fig. 9. The CDF of sharding consensus time.

server, so they can find idle servers in time. However, the distribution node is unable to view the queue length of shards, so it cannot distribute transactions according to the optimal queue length of each shard in real time. In the experiments, we found that the optimal queue length of the shards is an interval. Therefore, we look for appropriate

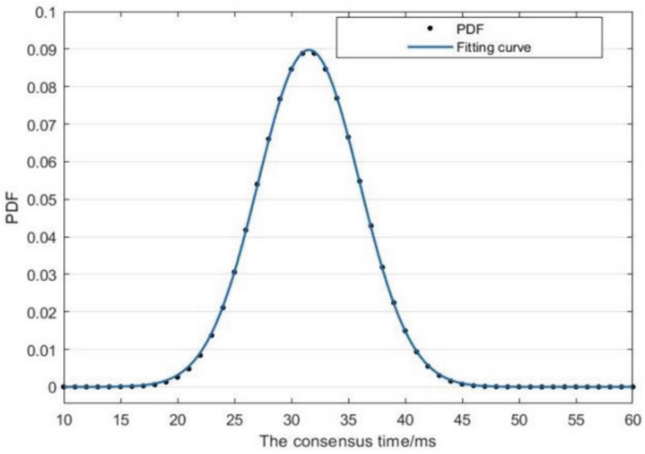

Fig. 10. The PDF of sharding consensus time.

R, C, and the service intensity ρ_i of each shard to obtain a large optimal queue length interval and a large social revenue.

Assuming that the optimal queue length of the shards $N_i \in [a_i, b_i]$, $0 < a_i < b_i$. The shards are divided into three states, (0, a_i, b_i), and the shard leaders can send status (0, start, stop) to the distribution node for real-time feedback while queue length is equal to 0, ai, bi. The distribution node stores the shards status in the local and changes it in time according to the shards feedback. The distribution node changes the target shards when it receives [stop] and reselects the target shards when it receives [start]. There are three cases:

1) When the shards state is all 0 or a_i, the distribution node sends transactions in order of the social revenue of shards.
2) When the shards state has 0 and a_i, the distribution node sends transactions to the shards with state 0 first.
3) When the shards state has ai and b_i, the distribution node sends transactions to the shards with state a_i in order of the social revenue of shards.

Assume that the service rate t_i,of the i-th shard obeys a Poisson distribution, i.e. $1/t_i$ transactions can be processed in one second. The optimal queue length interval is maximized by fixing R, C and varying ρ_i.

The average social revenue for each shard is greatest when $R = 2.5$ and $C = 10$, and the range of optimal queue length intervals and social revenue are calculated at this time, as shown in Table 3. From the nature of the Lambert W function [20], the solutions obtained at the 0^{th} branch are all true solutions, and the data that do not match have been removed when selecting the maximum social revenue.

TPS of Raft-S

The Raft-S consensus algorithm introduces the mechanism of social benefit, that is, the transaction pays the queuing fee in the slice and the reward after completing the consensus

Table 3. Optimal queue length intervals and social revenue of shards

shards	1/ti	Initial transaction arrival rates	Optimal queue length intervals	The social revenue
1	30	19	[3, 56]	30.22
2	40	27	[3, 88]	46.73
3	50	36	[3, 134]	64.29
4	60	45	[3, 186]	82.5

brings benefits to the whole system. When the reward is too high and the queuing fee is too low, the number of transactions entering the queue will greatly increase, causing system congestion. Through simulation, we find the appropriate queuing fee and reward, and fit the slice consensus efficiency to set the arrival rate to make the system smooth and obtain the optimal revenue.

When the system is stable, the average team leader in each segment is greater than 1, which means that the throughput of the whole system is the sum of the transactions processed per unit time of each segment, that is, the sum of the service rate. The system capacity and even throughput of traditional Raft can achieve linear growth. Assuming k slices, the average consensus efficiency for fragmentation is x, and y for conventional Raft. At this time, the throughput of Raft-S tends to be linear with conventional Raft, and its throughput is k times that of conventional Raft.

The distribution node forwards transactions at a rate of v1 and the sum of the transaction processing rates of the shards is v2, and that v1 is much greater than v2, this indicates that the Raft-S has sufficient transactions to the shards. If each shard is always running at high speed, the sum of the number of transactions processed per unit time by each shard is the TPS per unit time of the Raft-S. Taking Table 3 as an example, Raft-S can process 180 transactions in one second.

5 Conclusion

Raft is a strong leader algorithm, which puts the leader under greater pressure in communication, load, etc., and Raft's consensus efficiency decreases when the system size increases. Firstly, we propose a Raft-S consensus algorithm by combining the sharding technique, which ensures the global consistency and security of the sharding consensus while linearly increasing the Raft capacity and throughput. Secondly, we analyze the fragmentation consensus time and establish the queuing model applicable to Raft-S. Finally, in order to avoid the trading imbalance caused by network shard random distribution transactions, we design a trading allocation strategy applicable to the Raft-S consensus algorithm based on queuing theory, and fit the average time required for each shard to process an exchange. The distribution strategy makes the system achieve the highest social benefits while processing more transactions and improve the system TPS.

The cross-chain technology is a research hotspot for the application of the consortium blockchain. We will continue to study the cross-chain technology, such as sharding and side-chain and explore the evolution and implementation of the consortium blockchain.

References

1. Ongaro, D.: Consensus: bridging theory and practice. M.S. dissertation, Stanford University, Palo Alto, United states (2014)
2. Quorum. https://www.jpmorgan.com/global/Quorum
3. Cai, X., Deng, y., Zhang, L., et al.: The principle and core technology of blockchain. Chin. J. Comput. **44**(01), 84–131 (2021)
4. Sakic, E., Kellerer, W.: Response time and availability study of raft consensus in distributed SDN control plane. IEEE Trans. Netw. Serv. Manage. **15**(1), 304–318 (2018)
5. Tian, S., Liu, Y., Zhang, Y., et al.: A byzantine fault-tolerant raft algorithm combined with schnorr signature. In: 2021 15th International Conference on Ubiquitous Information Management and Communication, pp. 1–5. IEEE, Seoul (2021)
6. Huang, D., Li, L., Chen, B., et al.: RBFT: a new Byzantine fault-tolerant consensus mechanism based on Raft cluster. J. Commun. **42**(3), 209–219 (2021)
7. Gu, R., Huang, D.: A leadership transfer algorithm for the raft. In: Sun, Y., Cai, L., Wang, W., Song, X., Lu, Z. (eds.) CBCC 2022. CCIS, vol. 1736, pp. 13–30. Springer, Singapore (2022). https://doi.org/10.1007/978-981-19-8877-6_2
8. Baker, J., Bond, C., Corbett, J., Furman, J., et al.: Megastore: providing scalable, highly available storage for interactive services. In: 2011 - 5th Biennial Conference on Innovative Data System Research, Conference Proceedings, pp. 223–234. CIDR Press (2011)
9. Corbett, J., Dean, J., Epstein, M., et al.: Spanner: Google's globally distributed database. In: Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation, pp. 251–264. USENIX Association (2012),
10. Glendenning, L., Beschastnikh, I., Krishnamurthy, A., et al.: Scalable consistency in Scatter. In: Proceedings of the 23rd ACM Symposium on Operating Systems Principles, pp. 15–28. ACM Press (2011)
11. Zheng, P., Xu, Q., Zheng, Z., et al.: Meepo: sharded consortium blockchain. In: 37th IEEE International Conference on Data Engineering, pp. 1847–1852. IEEE Press (2021)
12. Xu, X., Sun, G., Luo, L.: Sharding algorithm based on evolutionary game in the IoT-blockchain. J. UESTC. **51**(3), 363–370 (2022)
13. Bai, S., Chen, M.: Research on hierarchical and sharding blockchain for industrial internet. Comput. Eng. 1–14 (2022). https://doi.org/10.19678/j.issn.1000-3428.0064338
14. Wang, M., Huang, J., Shao, X.: Reputation-based blockchain sharding consensus scheme. J. Comput. Sci. **49**(10), 297–309 (2022)
15. Yoo, H., Yim, J., Kim, S.: the blockchain for domain based static sharding. In: 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, pp. 1689–1692. IEEE Press (2018)
16. Kim, S.: Two-phase cooperative bargaining game approach for shard-based blockchain consensus scheme. IEEE Access **7**, 127772–127780 (2019)
17. https://blog.csdn.net/a1102325298/article/details/96481642
18. https://blog.csdn.net/dog250/article/details/72849985
19. https://baike.baidu.com/item/Little'slaw/10340863
20. Long, M., Zhou, T.: A Survey of Properties and Applications of the Lambert W Function. Hengyang Normal University Press, Hunan (2011)