Proceedings of the 2023 International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/27/20230173

Scheme for improving RAFT-based blockchain performance

Li Xiao[1, †], Songqiao Yang[2, †] and Tianxing Zhang[3, 4, †]

[1] Faculty of Business, Nankai University, Tianjin, China, 300071
[2] Department of Information science and Engineering, Shandong Normal University, Shandong, China, 250014
[3] Faculty of Information Technology, Beijing University of Technology, Beijing, China, 100022

[4]20072118@emails.bjut.edu.cn
[†]These authors contributed equally.

Abstract. Improving RAFT-based blockchain systems' performance and their relationship with environmental factors is a lack of concern in recent studies and is essential in the production environment. To attain this, it is necessary to analyze the performance, especially the blockchain system's throughput, latency, and robustness. The two most widely used RAFT-based platforms, etcd, and Hyperledger Fabric's evaluation, were conducted to discover the factors influencing the system's performance and promoting methods. The evaluation focused mainly on throughput, latency, and robustness, including evaluating the reading and writing process, changing the number of keys, connections, and clients in etcd, and comparing the process and the two platforms. The only number of clients significantly impacts etcd's performance, and etcd's performance is better than Hyperledger Fabric's. Besides, both two platform shows that reading performs better than writing. So, to improve the system's performance, controlling the number of clients and focusing on the writing process is the key.

Keywords: blockchain, RAFT, throughput, latency, robustness.

1. Introduction

Blockchain has attracted scholars' attention ever since its proposal. Data stored on blockchain is traceable, open, transparent and can't be tampered. Blockchain technology has been a hot topic worldwide in recent years. According to the World Economic Forum survey (Forum, 2015), blockchain will house 10% of the world's economy by 2027. Blockchain technology is also becoming more popular in academia. Three categories can be used to categorize blockchain research. The first is to do research on digital currencies based on blockchains, both centralized and decentralized. The second is the use of blockchain technology in settings other than digital currency, like managing medical information security. The third is the research behind blockchain technology. Researchers are beginning to understand that blockchain can be separated from virtual currency to develop ground-breaking technical frameworks in other fields. Some researchers have started looking into the underlying technologies, including smart contracts, scalable consensus methods, and difficulty management for mining. RAFT consensus algorithm as one of the consensus algorithms, applications for enterprise blockchain are

DOI: 10.54254/2755-2721/27/20230173

increasingly being used by businesses. The RAFT algorithm is commonly used by private blockchains, such as enterprise blockchains, to obtain consensus [1].

Double spending and the Byzantine Generals Problem are two issues that blockchain applications need to address. To address these issues, equivalent consensus algorithms must be developed, and distributed systems have long been the focus of research into consensus algorithms. On the blockchain, some portable consensus algorithms are used. In this section of the text, these consensus algorithms are thoroughly described. The PoW(Proof of Work) consensus algorithm, which is the consensus process employed in Bitcoin, was developed by Cynthia Dwork [2]. The fundamental concept is the distribution of accounting rights and benefits via mathematical competition amongst nodes. On the basis of the data from the preceding block, various nodes calculate a particular answer to a mathematical issue. It's challenging to solve mathematical puzzles. It's challenging to solve mathematical puzzles. The first node to finish this mathematical puzzle creates the following block and receives a specific amount of bitcoins in return. In order for someone to effectively safeguard the blockchain via PoW, he must own more than 50% of the world's computing power. Afterwards, QuantumMechanic put out POS (Proof of Stake), which was initially used with PPCoin. The concept of coinage exists in PoS for the digital currency [3].

A node acquires more network rights the longer it keeps a coin. Each node's hashing computation is constrained by PoS, and mining difficulty is inversely correlated with coin age. A blockchain using PoS is more secure as the value of the blockchain rises since an attacker would need to amass a significant amount of coins and hold them for an extended period of time to assault the block. Dan Larimer then put out the DPoS (Delegated Proof of Stake) consensus process, which is illustrated by BitShares [4]. In the DPoS consensus algorithm, the proper functioning of the blockchain relies on trustees (Delegates) that are fully equivalent. In the Bitcoin network, a trustee's node server functions as the equivalent of a miner, earning block rewards and transaction fees. The project sponsor chooses the trustee composition for a blockchain project, which is normally 101 trustees. Each cryptocurrency owner is eligible to take part in the trustee election process, including voting and running for office.

DPoS-based blockchains are more effective and power-efficient than PoW and PoS-based ones. PBFT (Practical Byzantine Fault Tolerance) was proposed by Miguel Castro and Barbara Liskov on the basis of this, and Byzantine fault tolerance may be a good way to deal with transmission problems in distributed systems [5]. Early Byzantine systems used exponential operations, while PBFT brought the complexity of the algorithm down to a polynomial level, greatly increasing efficiency. Lamport devised the Paxos method to resolve the consistency problem under particular circumstances after the Byzantine general problem was posed [6]. Ongaro proposed the RAFT algorithm, which divides server nodes into three states—leader, follower, and candidate—in order to set the server nodes into these three states—but it was not adopted because the paper's content needed to be more complex for people to understand and engineering implementation. For the duration of a semester, there is only one leader, who responds to all client queries [7]. In an RAFT-based system, only the leader distributes ledger entries to the other servers. Similar outcomes to Paxos are achieved by RAFT, but with simpler engineering implementation and comprehension. The RAFT algorithm is commonly used by private blockchains, including enterprise blockchains, to arrive at consensus [1].

Current research on the DRAFT algorithm includes 1) Research on how the DRAFT algorithm can be applied to various domains. According to Xu et al., their studies focus on wireless blockchain networks with malicious interference and the security performance of them. It gives analytical guidance for deploying wireless blockchain networks [8]. Hou et al. propose a method that can intelligently move transactions from busy regions to idle areas. This transaction migration scheme is used in IoT applications for private blockchain based on RAFT on it. And it can reduce the latency in high data flow circumstances [9]. 2) research on novel models for optimizing RAFT, such as Huang et al. noted that the lack of network stability would affect the blockchain performance and proposed an analytical model for the RAFT consensus algorithm. In theory, the analytical model is able to predict the time and probability of splitting the network as well as optimize the parameters based on the RAFT [10]. Liu et al. propose a port supply chain system architecture enabled by the alliance blockchain and a system verification framework for the port supply chain smart contract with probabilistic behavior [11]. It

DOI: 10.54254/2755-2721/27/20230173

refined the contract and enhanced its interaction with the outside environment to improve its performance. Experiments have proved that it can significantly shorten the time of processing tasks. K. Choumas and T. Korakis presented mathematical models to estimate the ranges resulting in the desired leadership probabilities [12]. 3) Research novel algorithms to improve DRAFT performance, such as works done by Wang et al., to enhance the RAFT algorithm's performance, the RAFT algorithm is proposed, and the "monitor" role is added to optimize the algorithm performance [13]. 4) Performance studies about RAFT algorithms like H. Howard and Malte Schwarzkopf's experiments of the RAFT authors were repeated, and the RAFT was evaluated [14]. D. Huang, X. Ma and S. Zhang investigates the performance of the adopted consensus algorithm RAFT in networks where the packet loss rate is not negligible [15]. Barger et al. described the design and implementation of Fabric's BFT sequencing service [16].

This paper studies the performance of the RAFT algorithm in a real environment through practical reality. To make the research closer to the real production environment, this paper chooses two widely used blockchain platforms based on the RAFT, etcd, and Hyperledger Fabric to analyze their performance. The paper's objective is to enhance the system's performance. First, this paper tests how number of connections, keys, connections, and clients will influence throughput and latency of the etcd cluster during both writing and reading process. The reading process can be divided into linearizable reading, and serializable reading. And then, this paper performs the robustness test of etcd and optimizes the performance by adjusting the disk priority. The results show that the number of connections, keys, linearizable reads, and serializable reads has little impact on the performance of the etcd and fabric clusters. The only variable that impacts the cluster performance is the number of clients, and the latency can be reduced to some extent by disk tuning.

The main issues addressed in this paper are Section 2 Preliminaries, Section 3 Methods, Section 4 Results, Section 5 Discussion and Section 6 Conclusion.

2. Preliminaries

This research mainly focuses on three indicators, they are:

- throughput: the number of tasks the system processes in a period (TPS).
- latency: The effective elapsed time of data from input to output.
- robustness: the ability of a system to survive under abnormal and dangerous conditions.

The test takes place on two platforms, etcd, and Hyperledger Fabric.

Etcd cluster

Etcd is a simple, secure, fast, and reliable distributed key-value store used by the Kubernetes cluster manager [17]. It is written in Go and uses the RAFT consensus algorithm to manage a highly-available replicated log. Its application scenario is mostly service registration and discovery. It can also be used for key-value storage. Applications can read and write data in etcd.

In this paper, an etcd cluster with three nodes is built on one PC using goreman tools. A commitment for the etcd cluster relies on the replication to a majority of nodes that are available. It can also prevent one or more nodes from failure [17]. In this case, the majority is 2. It reflects the robustness of the cluster. The research also used the official benchmarking tools to evaluate the cluster's throughput and latency. The relevant configuration of the system is as follows:

- Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz
- 1 machine(client) of 1 vCPU + 1024MB Memory + 8GB SSD
- Red Hat 10.3.1-1
- etcd 3.5.4, go 1.18.2

2.1. Hyperledger fabric based service

Fabric is a modular and extensible open-source system for deploying and operating permissioned blockchains and one of the Hyperledger projects hosted by the Linux Foundation [18]. It uses execute-order-validate architecture to reach an agreement, and the ordering service is based on the etcd library. The consensus used in fabric v2.x is RAFT, kafka, and solo.

DOI: 10.54254/2755-2721/27/20230173

The research used caliper, a performance testing tool, to evaluate the performance of querying all assets, creating assets, updating assets, and querying one asset processes. The relevant configuration of the system is as follows:

- AMD Ryzen 7 5800H with Radeon Graphics
- 1 machine(client) of 2 vCPU + 4GB Memory + 20GB SCSI
- Ubuntu 22.0.4.1
- fabric v2.4.7, ca v1.1.5, caliper v0.4.2

3. Methods

In this paper, two schemes are adopted to test the throughput and its relationship with the environmental factors of a RAFT-based blockchain system. One specific method is to test the throughput while changing the amounts of clients, connections, and keys using a benchmark. This evaluation took place in both the reading and writing data processes. Then by data collection and analysis, the research explored the relationship between performance and those environmental factors. The research adopts the idea of control variables. When testing one factor, the other two are control variables. And this part of the test is conducted on the etcd cluster due to the high performance based on the RAFT consensus.

The other method is to test the throughput by running a simple fabric application using Caliper with the Hyperledger Fabric environment because the Fabric supports a stronger configuration function and policy management function that could further enhance the flexibility and adaptability of the system. Transactions, including querying all assets, creating an asset, updating an asset, and querying an asset, the four processes will separately execute 8970, 5000, 1603, and 10177 times.

As for latency, the testing method is the same. The only difference is that etcd's benchmark can only show average latency, Caliper can also show the minimum and the maximum.

And for robustness testing, the research shuts one node down at a time on the etcd cluster to see if it can still work.

3.1. Testing the relationship between performance and parameters

The Detailed designed procedure is as follows:

- When changing the amounts of connections, the key number is 10000, client number is 1.
- When changing the amounts of clients, the key number is 10000, connection number is 1.
- When changing the amounts of keys, the connection number is 1, client number is 100.

The parameter above is for both the reading and writing data process. Other parameters include key size, value, target nodes, and linear and serializable reading differences. In all tests, the key size is eight, and the value is 256. The target node is the leader to simulate the real environment. Linearizable means a single operation on a single object in real-time order. Serializable means multi-operations on multi-objects in arbitrary total order. The research tests these two ways separately.

3.2. Testing the relationship between performance and parameters

The test result includes the following:

- The success and failure the number of the process.
- Send rate.
- Maximum, minimum, and average latency.
- Throughput.

This paper focuses on the average latency and the throughput.

DOI: 10.54254/2755-2721/27/20230173

4. Results

4.1. Testing the relationship between performance and parameters

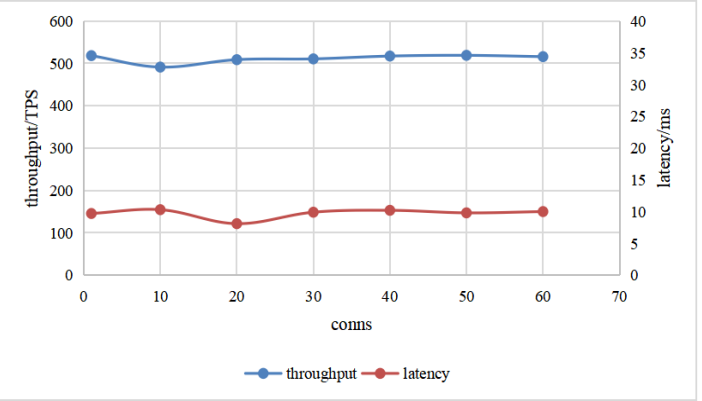

Figure 1. The variation of throughput and latency with number of connections in writing process.

Figure 1 reflects the relationship between connections and performance in the writing process. It shows that the connections won't affect the performance of the etcd cluster much in the writing process.

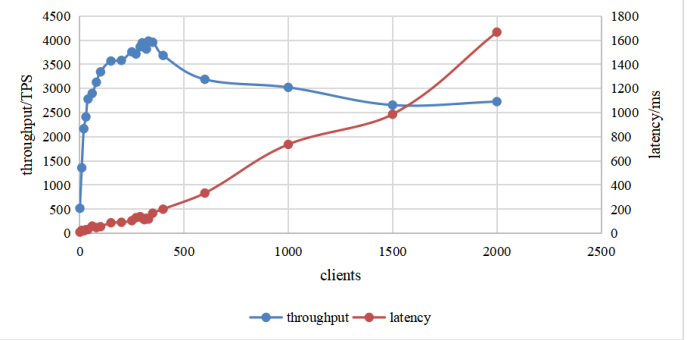

Figure 2. The variation of throughput and latency with number of clients in writing process.

Figure 2 reflects the relationship between clients and performance in the writing process. There is a linear relationship between the number of clients and the latency. However, when clients reach a certain amount, the throughput grows slowly or even drops a little. Therefore, clients are recommended to stay within this number.

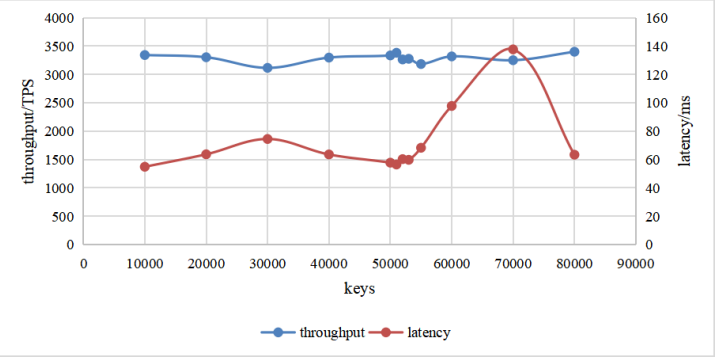

Figure 3. the variation of throughput and latency with number of keys in writing process.

DOI: 10.54254/2755-2721/27/20230173

Figure 3 reflects the relationship between keys and performance in the writing process. It shows that though throughput won't change much as the number of keys increases, the changes in latency are pretty irregular. It may be attributed to the cluster's ability to process data is limited. And as the data grew, the system became more unstable.

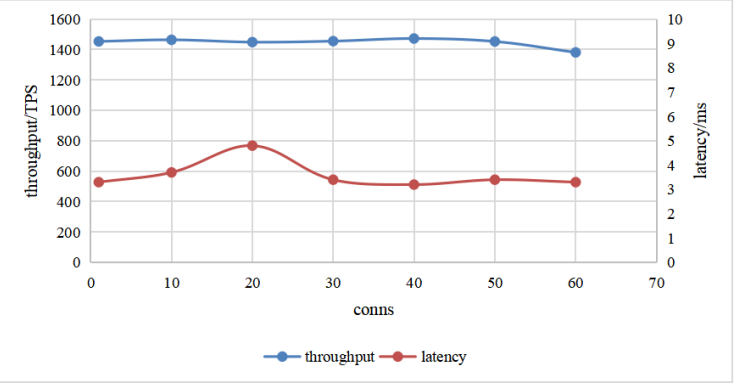

Figure 4. the variation of throughput and latency with number of connections in reading process.

Figure 4 reflects the relationship between connections and performance in the reading process. The throughput and latency remain stable as the connections increase as well.

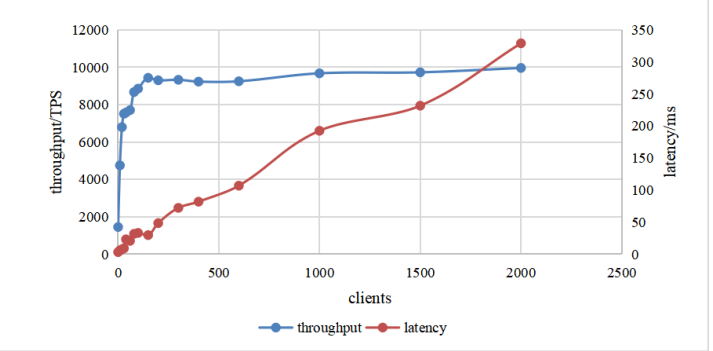

Figure 5. the variation of throughput and latency with number of clients in reading process.

Figure 5 reflects the relationship between clients and performance in the reading process. The changing trend is similar to the writing process, with a lower latency and a higher throughput.

Figure 4 and Figure 5 shows that the changing trend of the reading and writing process is the same, only the former with a lower latency and a higher throughput. Whether the process is reading or writing has nothing to do with the relationship between performance and environmental factors. And reading is faster than writing. The reason is that writing is much more complicated than reading.

Table 1. Fabric results analysis.

Name	Succ	Fail	Send rate (TPS)	Max latency(s)	Min latency(s)	Avg latency(s)	Through--put (TPS)
Create	5000	0	46.8	1.11	0.02	0.12	46.8
Change	1603	0	55.1	2.04	0.02	0.11	51.6
Query all	8970	0	309.2	0.09	0.00	0.02	309.1
Query 1	10177	0	350.8	0.09	0.00	0.01	350.7

Table 1 shows the performance of the fabric tested by the caliper. All the transactions succeeded. Creating an asset's throughput is 46.8, changing one is 51.6, query all is 309.1, and query one is 350.7.

It is easy to see from Table 1 that fabric is slower than etcd. It is easy to understand since applications on fabric contain a 6-step workflow, not simply an order process. But improving the etcd cluster's

DOI: 10.54254/2755-2721/27/20230173

performance can also work for a real system built on fabric because its order process is based on etcd. In addition, both etcd and fabric share the character that the performance of the reading process is significantly better than the writing process.

As for the reading process, the message here is that reading is divided into linearizable reading and serializable reading.

Table 2. Read performance analysis.

Keys	Key size	Value	conns	clients	ways	throughput	latency
10000	8	256	1	1	linearizable	1453.4129	3.3
10000	8	256	1	1	serializable	4013.8078	2.2
100000	8	256	100	1000	linearizable	6024.9124	464
100000	8	256	100	1000	serializable	8809.1451	282.5

Table 2 shows the differences in linearizable reading and serializable reading in the field of performance, and Serializable reading has a higher throughput and a lower latency than linearizable reading. So it seems that serializable reading's performance is better, but it is not read from legal nodes but from any node so that it may receive expired data.

4.2. The robustness of RAFT-based cluster

The etcd cluster can't provide service after killing two replicas. But it can still offer service after killing only one replica because the consistency principle of RAFT requires more than half of the replicas to agree on the same task. If half of them are killed, then there won't be more than half of them to reach an agreement.

5. Discussion

Based on the experiment, the throughput of the etcd cluster is most crucially impacted by the number of clients. And with the number grows, there will be a peek value. So, finding out what will impact the peak value is significant. Supposedly, it may be influenced by the number of nodes of the etcd cluster and the performance of the PC that carries this etcd cluster. Since all nodes have to hold the replicas, the process will undoubtedly be slower as the number of nodes increases, which leads to the peek dropping and appearing with fewer clients. But this study needs to carry on in a real muti-PC environment. The PC's performance that will affect the etcd cluster is easy to understand, but the key is to find out how and what (perhaps CPU, internal storage, and so on) exactly will affect the etcd cluster's throughput.

One method to improve the latency is disk tuning. Take etcd as an example; the etcd cluster is very sensitive to disk latency. Because etcd must persist the data to its log, the disk activity of other processes may cause a long fsync delay. Etcd may miss the heartbeat, resulting in request timeout and temporary leader loss. The method is to use the Best Effort policy and specify etcd's priority as 0, which is the highest. The result can be seen in table 3.

Table 3. Disk tuning result.

Keys	Key size	Value	conns	clients	ways	throughput	latency	Disk tuning
100000	8	256	100	1000	leader	2437.9109	853.2	yes
100000	8	256	100	1000	leader	2402.9260	929.9	no

Table 3 shows that disk tuning can lower the latency to a certain extent, but not significantly. For further tuning, the split-disk strategy may be useful. Specifically, Split snapshot and wal on two SSD disks to improve the overall IO efficiency. Snapshot and wal are two storage directories of etcd. They write in different ways. Snapshot is a direct dump file in internal storage, while wal is written sequentially. The system tuning methods are different in these two ways. Snapshot can be improved by increasing the IO smooth write to enhance the disk's IO ability, while wal can be improved by reducing

DOI: 10.54254/2755-2721/27/20230173

the page cache to make the writing timing earlier. Therefore, separating snapshot and wal and improving the efficiency separately can improve the performance of etcd.

Other methods like network tuning may also improve RAFT-based blockchain's throughput or latency. But the optimization won't be striking since the experiment occurred on one PC. External factors like networks have little effect, so the experiment must be carried on a real muti-PC environment.

6. Conclusion

The consensus view of users toward tending to choose a blockchain system is that its performance is acceptable. This paper is carried out to test the performance of etcd and fabric, and this paper finds that:

● The correct number of clients is the most crucial for improving the etcd cluster

Through experiments, it is found that the number of connections and key values has little impact on the clients, while the number of clients significantly impacts the performance of the etcd cluster. And it is much more useful than disk tuning, network tuning, and other environmental optimizations.

Therefore, it is recommended that the number of clients should be at most this number. In addition, since the ordering process of fabric is based on etcd, the method of etcd cluster performance is also applicable to real systems built by fabric.

● Focus more on the reading process when testing the blockchain system’ s performance while improving it.

According to the research, the etcd and fabric platforms show a distinguishing feature: reading is much faster than writing. It is because the writing process has a more complicated and wider variety of operations. This is why when improving the performance of a blockchain system, the writing process should be paid more attention to.

These findings have important implications for the generalization of blockchain systems. This is because the super ledger structure is the most popular research and practical production platform. And the order process is based on etcd.

The paper’s subsequent work: 1) Study factors related to the inflection point of throughput with the number of customers (number of cluster nodes, system-related configuration). 2) build a real environment using multiple machines to study the difference in performance between an etcd cluster in a real production environment and an etcd cluster built on a single machine.

References

[1] Nguyen, Giang-Truong, and Kyungbaek Kim. "A survey about consensus algorithms used in blockchain." Journal of Information processing systems 14.1 (2018): 101-128.

[2] Moni Dwork and Cynthiaand Naor, "Pricing via processing or combatting junk mail" in Advances in Cryptology - CRYPTO' 92, Berlin, Heidelberg:Springer Berlin Heidelberg, pp. 139-147, 1993.

[3] P. Vasin, BlackCoin’ s Proof-of-Stake Protocol v2, 2014, [online] Available: https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf.

[4] T. Y. Song and Y. L. Zhao, "Comparison of blockchain consensus algorithm", Comput. Appl. Softw., vol. 25, no. 8, pp. 1-8, 2018.

[5] M. Castro and B. Liskov, "Practical Byzantine fault tolerance", Proc. Symp. Operating Syst. Design Implement., pp. 173-186, 1999.

[6] Lamport L.: The part-time parliament. ACM Transactions on Computer Systems16(2), 133-169(1998)

[7] Ongaro D, Ousterhout J (2013) In search of an understandable consensus algorithm (extended version)

[8] H. Xu, L. Zhang, Y. Liu and B. Cao, "RAFT Based Wireless Blockchain Networks in the Presence of Malicious Jamming," in IEEE Wireless Communications Letters, vol. 9, no. 6, pp. 817-821, June 2020, doi: 10.1109/LWC.2020.2971469.

[9] Hou, Lu, et al. "An intelligent transaction migration scheme for RAFT-based private blockchain in Internet of Things applications." IEEE Communications Letters 25.8 (2021): 2753-2757.

DOI: 10.54254/2755-2721/27/20230173

[10] Huang, Dongyan, Xiaoli Ma, and Shengli Zhang. "Performance analysis of the raft consensus algorithm for private blockchains." IEEE Transactions on Systems, Man, and Cybernetics: Systems 50.1 (2019): 172-181.

[11] Liu, Yang, et al. "Verifying the smart contracts of the port supply chain system based on probabilistic model checking." Systems 10.1 (2022): 19.

[12] K. Choumas and T. Korakis, "On Using Raft Over Networks: Improving Leader Election," in IEEE Transactions on Network and Service Management, vol. 19, no. 2, pp. 1129-1141, June 2022, doi: 10.1109/TNSM.2022.3147958.

[13] Wang, Y., Li, S., Xu, L., Xu, L. (2021). Improved Raft Consensus Algorithm in High Real-Time and Highly Adversarial Environment. In: Xing, C., Fu, X., Zhang, Y., Zhang, G., Borjigin, C. (eds) Web Information Systems and Applications. WISA 2021. Lecture Notes in Computer Science(), vol 12999. Springer, Cham. https://doi.org/10.1007/978-3-030-87571-8_62

[14] H. Howard, M. Schwarzkopf, A. Madhavapeddy and J. Crowcroft, "Raft refloated: Do we have consensus?", ACM SIGOPS Oper. Syst. Rev., vol. 49, no. 1, pp. 12-21, 2015.

[15] D. Huang, X. Ma and S. Zhang, "Performance Analysis of the Raft Consensus Algorithm for Private Blockchains," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 50, no. 1, pp. 172-181, Jan. 2020, doi: 10.1109/TSMC.2019.2895471.

[16] A. Barger, Y. Manevich, H. Meir and Y. Tock, "A Byzantine Fault-Tolerant Consensus Library for Hyperledger Fabric," 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Sydney, Australia, 2021, pp. 1-9, doi: 10.1109/ICBC51069.2021.9461099.

[17] Vohra, D. (2016). Installing Kubernetes on a Multi-Node Cluster. In: Kubernetes Microservices with Docker. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-1907-2_14

[18] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... & Yellick, J. (2018, April). Hyperledger fabric: a distributed operating system for permissioned blockchains. In Proceedings of the thirteenth EuroSys conference (pp. 1-15).