Journal of Information Systems Engineering and Management
2025, 10(1s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Improving the Scalability of Blockchain Based on Trust Practical Byzantine Fault Tolerance Algorithm (tPBFT)

Noor Sabah [1*], Sura F. Ismail [2], Dheyaa Jasim Kadhim [3], Sara S. Jawad [4]

[1] *Department of Electrical Engineering, College of Engineering, University of Wasit, Wasit, Iraq.*

[2] *Department of Informatics management System, College of Informatics Business, University of Information Technology and Communications, Baghdad, Iraq.*

[3] *Department of Electrical Engineering, College of Engineering, University of Baghdad, Baghdad, Iraq.*

[4] *Youth and Sport Ministry, Baghdad, Iraq.*

** Corresponding e-mail: noors@uowasit.edu.iq, dheyaa@coeng.uobaghdad.edu.iq, sura.fawzi89@uoitc.edu.iq, s.jawad12@yahoo.com*

ARTICLE INFO

Received: 30 Sept 2024
Revised: 25 Nov 2024
Accepted: 12 Dec 2024

ABSTRACT

Since Bitcoin's inception, the non-custodial financial and programmable aspects of blockchain technology have garnered significant global attention, akin to other groundbreaking innovations. Blockchain is a transparent and decentralized ledger that prioritizes immutability, transparency, decentralization, and privacy. Its potential extends to large-scale, beneficial applications across various sectors, encompassing healthcare, supply chain management, logistics, the Internet of Things (IoT), and more. Many industrial applications make use of permissioned blockchains. However, blockchain still faces inherent and external challenges. While permissioned blockchains are suitable for many use cases, they do have limitations, especially in terms of scalability and throughput.

In this paper, the authors propose implementing the proposed method by Swathi and Venkatesan using tPBFT (trust Practical Byzantine Fault Tolerance) instead of PBFT (Practical Byzantine Fault Tolerance) to improve the scalability and performance of the blockchain. The Hyperledger Fabric framework is conducted for the proposed method within the assessment of scalability, considering varying transaction volumes, and it demonstrates an enhancement in scalability. This work addresses the need for blockchain systems to evolve and adapt to meet the demands of various industries and applications, acknowledging that scalability is a key factor in their success.

Keywords: Blockchain, Improve Scalability, PBFT, tPBFT, Consensus mechanisms, Nodes, Hyperledger fabric.

INTRODUCTION

In traditional financial systems, every transaction necessitates verification by a third party. Banks and other financial institutions are responsible for verifying transactions when customers want to use their credit or debit cards to purchase anything at a marketplace. Even in cases where a buyer opts for a cash payment, the process necessitates withdrawing money from a bank, emphasizing that a third party always plays a role in the validation or verification of the transaction. The fundamental goal of the technology of blockchain is to create a decentralized infrastructure. Hence, the participation of a third party tends to centralize all transactions thus an enduring risk of a single point failure is introduced. This can be achieved through either a permissionless or a permissioned approach, both of which aim to create a decentralized infrastructure. In the case of cryptocurrency, therefore, public or permissionless blockchain is utilized whereby any person may take part in executing transactions. On the other hand, permissioned blockchain networks allow the network to choose certain individuals who can take part in block validation process. Such networks are often utilized within secure networks or private enterprises. This methodology results in the establishment of a distributed ledger that records transactions comprehensively, maintaining a detailed history of

each verified transaction. Additionally, it provides a shared platform that allows users to observe each other's transactions without the need for third-party intermediaries. Furthermore, this blockchain technology upholds the anonymity of all user data and transactions, ensuring that each system user possesses a copy of the perpetually expanding ledger [1]. In a permissioned blockchain, every participating peer is tasked with executing each transaction, maintaining a ledger, and engaging in the consensus process, which includes fault-tolerant mechanisms. However, this approach has limitations, particularly when it comes to supporting private transactions with secret contracts. Hyperledger Fabric stands out as among the best choices for providing a secure and adaptable foundation for industrial blockchains, it often utilizes the Practical Byzantine Fault Tolerance (PBFT) consensus algorithm. Despite the numerous advantages of blockchain technology, certain factors have held back its full-scale adoption in various industries. One of the prominent challenges is scalability. Scalability is referring to the ability of a system to handle increased volume of the work and accommodate its expansion. It is a crucial concern for organizations looking to leverage blockchain technology. As industries seek to integrate blockchain into their operations, addressing scalability becomes a vital item on their priority list [2, 3].

The choice to implement trust Practical Byzantine Fault Tolerance (tPBFT) over PBFT in this article was driven by a specific performance advantage, when the network expands to include more than 30 nodes, tPBFT demonstrates significantly improved performance compared to PBFT, this improvement is particularly evident in areas such as reduced node communication overhead, enhanced consensus effectiveness, and increased scalability.

(Croman et al., 2016) [4] explored how Bitcoin's current distributed overlay architecture is limited in its ability to support much greater throughputs and reduced latencies by critical and arbitrary constraints. Their findings suggest that block size and interim reparameterization be considered as a first step toward achieving high-load blockchain norms for the future. Additionally, significant progress will necessitate a thorough reevaluation of specialist methodologies, this strategy is in the process of developing a well-structured perspective on the design space for such situations.

(Wang et al., 2017) [5] shown how challenging it is to scale permissioned blockchain apps to effectively service a large number of clients without any problems. It gives an explanation of Blockbench, the framework for analyzing private blockchains. The author provides a fair way for correlating for various phases and facilitates a deeper understanding of other frameworks.

(Tuan et al., 2017) [6] They carried out a Blockchain evaluation that assessed such parameters as throughput, resilience, latency, and scalability to internal faults, their study included a comprehensive evaluation of Hyperledger Fabric, Parity, and Ethereum, three of the most famous blockchains that are private. Their results indicate that these frameworks are still some distance away from replacing traditional database frameworks commonly used for conventional data management, furthermore, the assessment of these three frameworks in terms of structural decisions at different stages of the blockchain product stack reveals issues related to their execution.

(Baliga et al., 2018) [7] established an evaluation technique in which they exposed Fabric to various workload configurations to examine its throughput and latency. The authors modify various transaction and chain code settings and investigate how they affect latency using a series of lower size benchmarks designed specifically for Fabric.

(Gorenflo et al., 2019) [8] rebuilt the Hyperledger Fabric permissioned blockchain infrastructure to increase exchange capacity from 3,000 to 20,000 trades per second. They adjusted the parameters governing computation and I/O overhead, focusing on resolving the execution bottlenecks that are currently present in Hyperledger Fabric.

The rest of this study is structured as follows: Section 2 provides the policies and procedures, Section 3 describes the Hyperledger Fabric v2.0, Section 4 presents the suggested method, Section 5 discusses the findings and analyses, and Section 6 conclude the article.

POLICIES AND PROCEDURES

In this section, we provide definitions for key terms related to this work:

- **Consensus:** These methods ensure the integrity and correctness of records in a distributed ledger. Consensus is a fundamental aspect of a distributed ledger; it is responsible for validating the legitimacy of all transactions and maintaining a consensus on the current state of the ledger within the network [9].
- **Proof of Work (PoW):** This is the consensus mechanism employed by public blockchains, such as the one used by Bitcoin. Various other consensus methods are available, including Practical Byzantine Fault Tolerance (PBFT), Proof of Stake (PoS), Proof of Authority (PoA), Proof of Capacity (PoC), and many others.

- **Smart contracts:** These digital counterparts of real-world contracts store a computer program within the blockchain. When funds are received, a smart contract acts as an escrow, releasing the funds upon the completion of specific objectives. For instance, in project execution, sponsors can transfer funds to the smart contract. If the project reaches its funding goal, the smart contract disburses funds to the project's author. Conversely, if the project doesn't raise enough money within the designated timeframe, funds are automatically returned to the backers. The immutability and distribution of the blockchain containing the smart contract ensure its reliability. The system can define various functions based on business logic [9, 10].

HYPERLEDGER FABRIC V2.0

The platform known as Hyperledger Fabric is designed to facilitate distributed record arrangements with a focus on privacy, robustness, flexibility, and versatility. It operates within a private framework and aims to simplify the complexity and unpredictability found in the financial ecosystem while supporting the use of pluggable components. It is currently operating version 2.x. and recent years have witnessed updates to Hyperledger Fabric. In a blockchain network, a limited number of nodes collaborate to process transactions. In the case of Hyperledger Fabric, which operates as a permissioned network, nodes are assigned unique identities by the Membership Service Provider (MSP). These nodes can run on physical hardware, containers, or virtual machines. Hyperledger Fabric categorizes nodes into three types: peers, orderers, and clients. Notably, there has been a significant change regarding peers in Hyperledger. The peers have been categorized into three groups: endorsers, consenters, and committers. Peers are responsible for executing and maintaining transactions within the ledger. Peers in the network serve multiple roles within the Hyperledger Fabric ecosystem. They receive ordered state updates in the form of blocks from the ordering service and store these blocks in the ledger. By default, all peers also function as committers, committing transactions to the ledger. Additionally, peers take on the role of endorsers, executing smart contracts and simulating transactions to validate them. Consenters are responsible for verifying whether peers have exchanged assets correctly. Orderers, on the other hand, manage the ordering of transactions, and the collection of orderers is referred to as the ordering service. Finally, end-users act as clients, sending transaction requests to peers. Clients coordinate orders and committers during the verification process [11].

3.1 Hyperledger Fabric Architecture Overview

In the network illustrated in Figure 1, three organizations (O1, O2, and O3) collaborate to establish a distributed ledger. Each of these organizations acts as a validating peer. Among the validating peers, O1, in this case, holds the role of the network initiator. Clients (C1, C2, C3) transmit transaction requests to validating peers, who then validate and broadcast the transactions. Peer node P1 is responsible for maintaining a copy of ledger L1, which is connected to C1. In a similar manner, peer node P2 maintains a copy of ledger L2 connected to C2, and peer node P3 replicates ledger L3 connected to C3. Channels, such as C1 and C2, are managed by channel configurations (CC1 and CC2, respectively), with O1, O2, and O3 having control over the channels according to policy guidelines. The network also includes an order placement service that serves as a network management hub and operates using the system channel. Every organization operates its own designated Certificate Authority (CA), and these CAs are responsible for delivering certificate services to their respective counterparts [11].

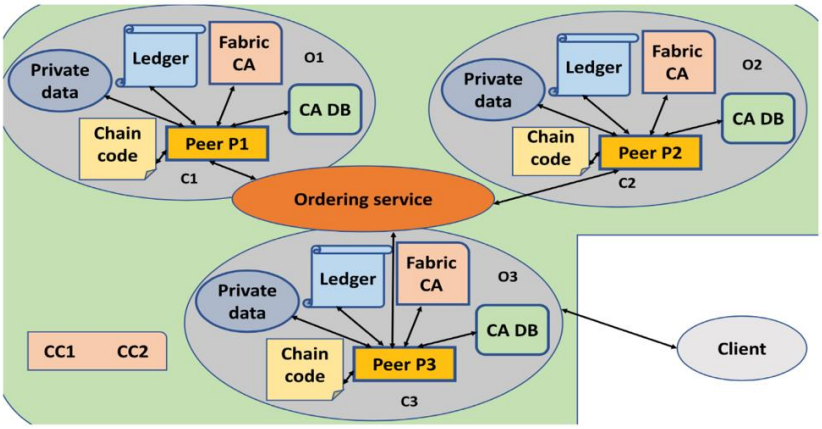

Figure (1) Overview of the Hyperledger Fabric system [2].

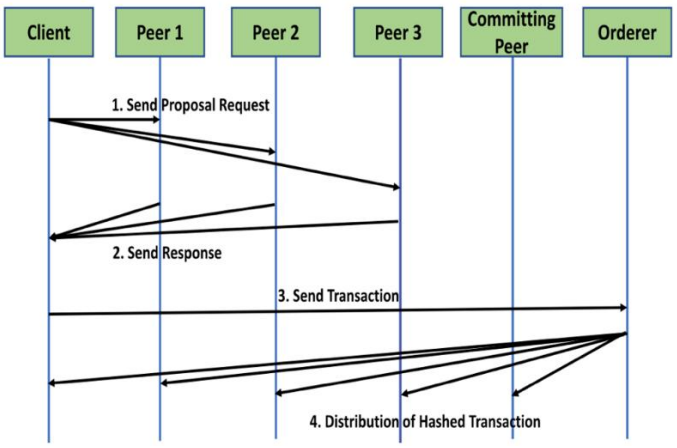

Figure (2) The transaction flow within the Hyperledger Fabric system [2].

To facilitate the transaction flow within Hyperledger Fabric and maintain data confidentiality, a configuration involving three endorsing peers and one committing peer is utilized, as depicted in Figure (2). The process unfolds as follows:

- **Proposal Request:** The client application initiates a request to trigger a chaincode function. This request is then transmitted to endorsing peers that are affiliated with authorized organizations.
- **Endorsement:** The endorsing peers receive the proposal request, replicate the transaction, and temporarily store sensitive data. They then provide the client application with the proposal response. The response includes the supported read/write set and a hash of the keys involved in the transaction.
- **Transaction Submission:** The client application sends the transaction, along with its hash, to the ordering service.
- **Block Distribution:** The ordering service adds the hashed transaction to a block and disseminates this block across the network's peers.
- **Data Validation:** During the commit phase, the peers validate the data by determining if they have permission to access it.
- **Data Verification:** Peers also verify whether their data has been received in the temporary data store. If they have the necessary permissions, they access and validate the data; otherwise, they collect and evaluate it from their peer counterparts.
- **Data Transfer:** After successful verification, a copy of the material is transferred to private storage for safekeeping, and the data in temporary storage is destroyed. This process ensures the confidentiality of sensitive data while allowing for the secure execution of transactions in the Hyperledger Fabric network [12].

THE PROPOSED METHOD

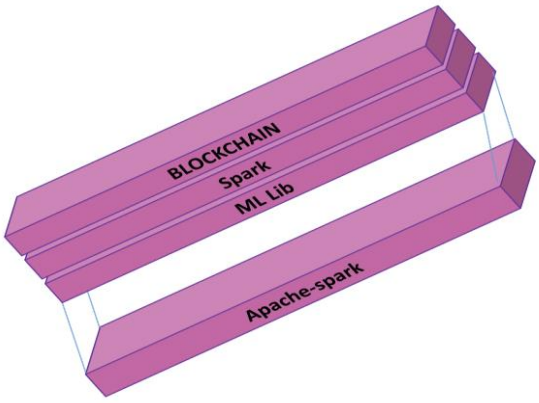

Figure (3) The proposed method [2].

The architecture depicted in Figure (3) serves as a valuable tool for assessing scalability according to latency and transaction throughput. Scalability is achieved when either the throughput increases or remains consistent as the number of transactions grows, or when latency remains constant or decreases with an increasing number of transactions [12]. Scalability refers to a system's ability to handle increasing workloads and growing demands. Users can choose to implement this suggested framework based on their specific business use cases. By altering at least one independent variable and measuring dependent components, the framework's performance can be evaluated to assess its scalability. Real-world production trials involve adjusting measurements, and ongoing studies may see changes to some metrics. Distributed machine learning (ML) can provide a solution to address scalability challenges.

The term "distributed machine learning" refers to an approach designed for multi-node systems that enhances their efficiency, capacity, and accuracy, particularly for larger input data sets. Distributed machine learning algorithms can effectively handle extensive data sets, meeting accuracy and computational requirements while providing scalability [13]. Scalability challenges can be addressed by leveraging Apache Spark MLlib, an open-source cluster processing system designed for real-time data processing. The primary advantage of Apache Spark lies in its in-memory cluster computing, which significantly accelerates various applications. Spark offers a programming interface for large-scale data parallelism and fault tolerance for non-critical failures. It is versatile, capable of handling various workloads, including streaming, iterative computations, batch applications, and complex queries. Figure (3) illustrates how blockchain has been integrated with Apache Spark, providing a comprehensive standard for multiple transactions and enhancing workflow efficiency. This work integrates runtime code and memory management outside of the Java virtual machine (JVM) to optimize the efficiency of SQL operations and data. Certain aspects of transaction validation can be parallelized using Spark. Fabric's transaction validation service has been revamped to incorporate Spark, enabling the parallelization of as many validations as feasible [14]. Blockchain comprises four layers: the application layer, network layer, consensus layer, and contract layer. Figure (4) demonstrates how Spark interacts with the application and consensus layers of the blockchain. When a transaction proposal is received, the order determines whether the client has authorization to submit the transaction. The individual transaction requests are then sent by the order to the Kafka cluster, where each fabric is mapped to Spark to create transaction orders. Before mapping, the Random Forest machine learning algorithm is applied to transactions due to its quick learning curve and high accuracy. This assists transactions in making channel decisions, enabling simultaneous execution. Ultimately, these transactions are consolidated into a block based on the maximum number allowed in a block [15], creating a distinct collection of transactions. Figure (5) illustrates the fabric and Spark mapping process along with the transaction workflow. The scalability of the system is clear, as extra transactions may be accommodated due to the simultaneous execution process.

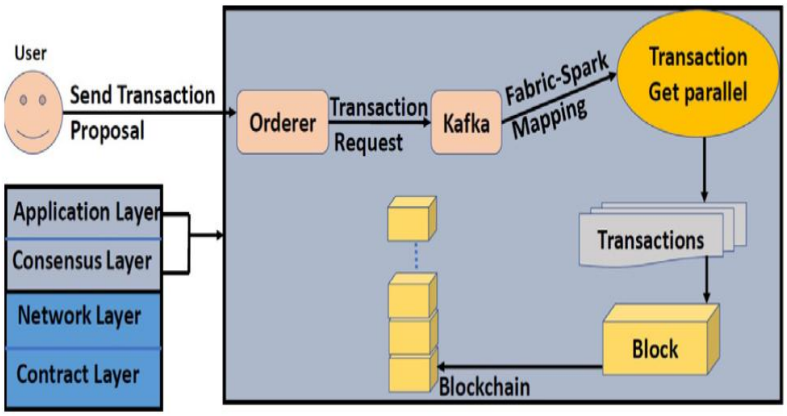

Figure (4) The blockchain's structure within the proposed method [2].

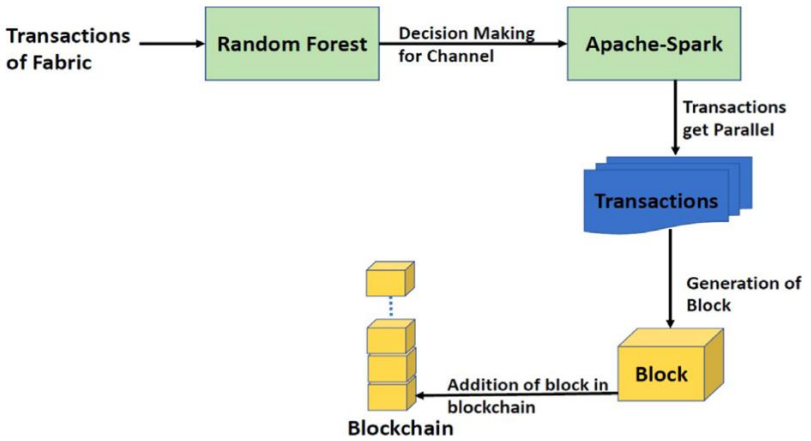

Figure (5) The transaction workflow as well as the mapping procedure between Fabric and Spark [2].

RESULTS AND ANALYSIS

A private mini blockchain network was established at NIT Karnataka to evaluate and validate the proposed solution directive. This network was built using Hyperledger Fabric, which was run on a 64-bit Ubuntu operating system with 8 GB of RAM. The most recent versions of the blockchain framework and Hyperledger Fabric using tPBFT were used to conduct the testing, the versions used were 1.4.0 and 2.0, respectively. The assessment considered three different companies. A Certificate Authority client (CA), a Membership Service Provider (MSP), and a peer were included in each of the N organisations that made up the network. As shown in Figure (1), a single channel connected all the companies. The experiments' chain code was developed in the Golang programming language. Hyperledger Fabric offers two distinct ordering methods: SOLO and a Kafka-based ordering service. Since SOLO is primarily for testing and not intended for real-world use, the tests in this work employed Kafka. The ordering service comprised a configurable number of Zookeeper nodes and Kafka servers. ZooKeeper, a distributed program, enables efficient synchronization and shared data management among nodes. To avoid split decisions, the number of Zookeeper nodes must be odd. For trust fault tolerance, a minimum of four Kafka servers was recommended. The system's hubs collaborated to form the blockchain network, with each hub's structure, design, and hardware differing as needed [16]. The test configuration included hubs responsible for performance evaluation, which served as clients capable of performing two types of tasks: load-generating clients and monitoring clients. Load-generating clients submitted transactions on behalf of end-users, while monitoring clients could query their peers and receive transaction progress updates [15]. The test also acquired and analysed the datasets required to evaluate performance indicators. Figure (6) displays the relationship between the number of transactions and confirmation time.

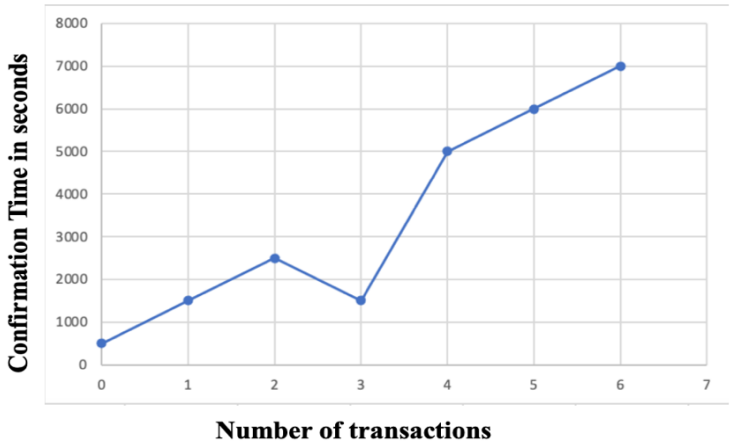

Figure (6) The relationship between the number of transactions and confirmation time.

We accept transactions in the range of 0 to 50000 as shown in Figure (6). Seconds have been measured as the confirmation time. It is evident that the confirmation time has been essentially constant for 30000 transactions. The confirmation times then gradually get longer as the number of transactions rises above 30000 transactions. These numbers were obtained from the experimental set-up described above. The results are for transactions beyond 30000, confirmation times dramatically rise. The system's throughput rises as the number of transactions rises, but at the expense of a rise in network latency. The overhead approach for extracting more transactions into blocks becomes more complicated as the number of transactions increases.

Figures (7) & (8) display throughput before and after proposed solution implementation using tPBFT algorithm respectively.

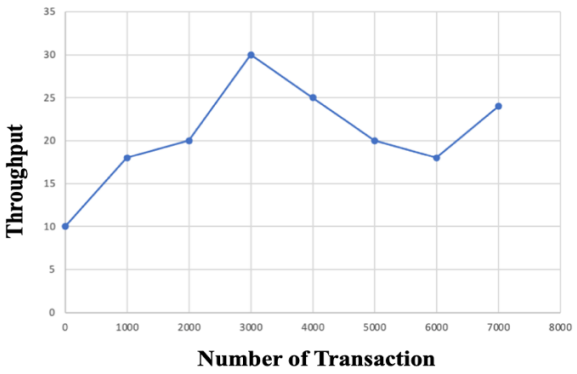

Figure (7) Throughput before proposed solution implementation.

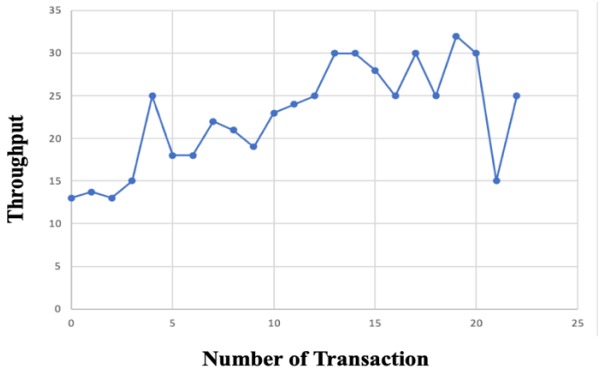

Figure (8) Throughput after tPBFT proposed solution implementation.

The throughput of a blockchain infrastructure is determined by the number of validated transactions executed per second. Most contemporary payment systems can typically handle an average of around 2,000 transactions per second. Hyperledger Fabric systems typically accomplish a throughput of 3,000 transactions per second, whereas blockchain systems based on Bitcoin only manage an average of seven transactions per second. Increasing the block size and network transaction demand can lead to hard forking in a blockchain. Although one way to enhance throughput in Hyperledger Fabric is to increase the block size, doing so poses risks to the decentralization and security of the blockchain, making it an unrealistic solution. In line with the proposed solution direction, Apache Spark was used to address this challenge. Figures (7) and (8) clearly demonstrate that the proposed framework significantly decreases the workload on Fabric. The results obtained were as follows: The Hyperledger Fabric demonstrates robust transaction throughput up to 30,000 transactions. The proposed solution using tPBFT algorithm increased the blockchain's throughput to more than 30,000 transactions. The inclusion of additional transactions in the system was facilitated by the implementation of parallelism in the solution directive, which simultaneously reduced overhead.

Figures (9) & (10) display latency before and after proposed solution implementation using tPBFT algorithm respectively.

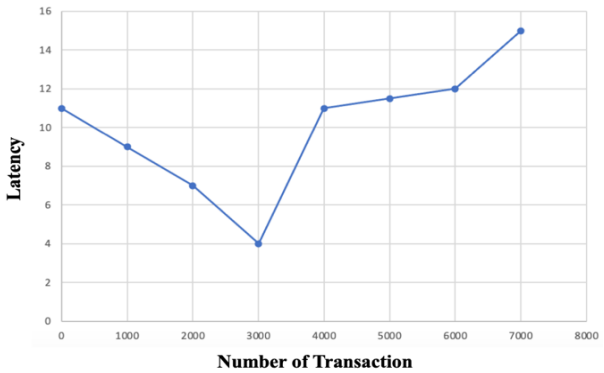

Figure (9) Latency before proposed solution implementation.

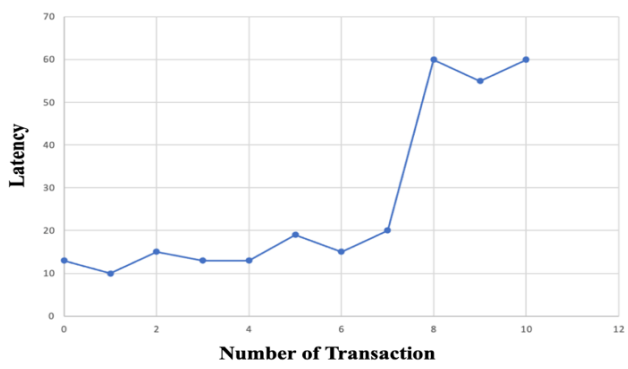

Figure (10) Latency after tPBFT proposed solution implementation.

The latency of a blockchain network is influenced by the time required for blocks to propagate across the network, with transaction verification time directly affecting network latency. Faster transaction confirmation times result in lower latency and faster network transmission.

PBFT experiences increased latency when there are more than 30 nodes, while tPBFT performs better at reducing latency. The following details become clear when analyzing and comparing the data in Figures 9 and 10: Up to 30,000 transactions, Hyperledger Fabric demonstrates a reasonable transaction latency; however, the latency abruptly increases beyond this point. The growing block size may be contributing to the increase in latency, as it implies that as the volume of transactions on the system increases, so does the delay. Network congestion resulting from increased network capacity or the need for additional resources to propagate larger blocks can also lead to higher latency. These findings emphasize the critical role of throughput, scalability, and latency in the efficiency of blockchain systems and offer insights into how different factors impact the efficiency of these systems.

The findings mentioned above clearly demonstrate that the implementation of the suggested framework by using tPBFT effectively maintains stability in terms of both throughput and latency, even when dealing with up to more than 30,000 transactions. Spark's capacity to parallelize the transaction processing significantly contributed to handling more transactions, resulting in improved scalability. This increased scalability is also reflected in the reduced confirmation time. The use of Spark as the foundation for the blockchain allowed the consensus mechanism to efficiently handle a greater volume of transactions. Consequently, the consensus mechanism in Hyperledger Fabric using tPBFT exhibited superior scalability performance, whereas in PBFT, scalability would start to decline when the number of nodes exceeded 30. The test findings clearly demonstrate improvements in characteristics like as scalability, transaction latency, transaction throughput, , and confirmation time compared to their baseline performance.

CONCLUSIONS

Over the past decade, blockchain technology has made rapid advancements, bringing us closer to its integration into everyday life. The adoption of blockchain technology has steadily increased, attracting a growing number of users. However, despite its progress, blockchain's performance still requires significant improvements, especially when compared to mainstream CPUs. The current architecture often encounters system bottlenecks, prompting specialists to carefully consider solutions to the scalability challenge. The proposed method using tPBFT uses distributed machine learning techniques to handle scalability issues in an innovative way, hence tackling these challenges. Notably, the suggested technique using tPBFT performs effectively even when the blockchain system comprises more than 30 nodes and experiences a higher volume of transactions.These advancements indicate a promising direction for enhancing blockchain technology and making it more robust and scalable for various applications and user needs.

REFERENCES

[1] S. S. Hazari and Q. H. Mahmoud, “Improving transaction speed and scalability of blockchain systems via parallel proof of work,” Futur. Internet, vol. 12, no. 8, 2020, doi: 10.3390/FI12080125.

[2] P. Swathi and M. Venkatesan, “Scalability improvement and analysis of permissioned-blockchain,” ICT Express, vol. 7, no. 3, pp. 283–289, 2021, doi: 10.1016/j.icte.2021.08.015.

[3] S. Tang, Z. Wang, J. Jiang, S. Ge, and G. F. Tan, “Improved PBFT algorithm for high-frequency trading scenarios of alliance blockchain,” Sci. Rep., vol. 12, no. 1, pp. 1–12, 2022, doi: 10.1038/s41598-022-08587-1.

[4] K. Croman, C. Decker, I. Eyal, A.E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Sirer, D. Song, R. Wattenhofer, On scaling decentralized blockchains, 9604, ISBN: 978-3-662-53356-7, 2016, pp. 106–125, http://dx.doi.org/10.1007/978-3-662-53357-4_8.

[5] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K. L. Tan, “BLOCKBENCH: A framework for analyzing private blockchains,” Proc. ACM SIGMOD Int. Conf. Manag. Data, vol. Part F1277, pp. 1085–1100, 2017, doi: 10.1145/3035918.3064033.

[6] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, “Untangling Blockchain: A Data Processing View of Blockchain Systems,” IEEE Trans. Knowl. Data Eng., vol. 30, no. 7, pp. 1366–1385, 2018, doi: 10.1109/TKDE.2017.2781227.

[7] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat, and S. Chatterjee, “Performance characterization of hyperledger fabric,” Proc. - 2018 Crypto Val. Conf. Blockchain Technol. CVCBT 2018, pp. 65–74, 2018, doi: 10.1109/CVCBT.2018.00013.

[8] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, “FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second,” ICBC 2019 - IEEE Int. Conf. Blockchain Cryptocurrency, pp. 455–463, 2019, doi: 10.1109/BLOC.2019.8751452.

[9] D. Ongaro, J. Ousterhout, In search of an understandable consensus algorithm, in: USENIX Annual Technical Conference, 2014.

[10] A. Bessani, J. Sousa, and M. Vukolić, “A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform (Short Paper),” Ser. 2017 - 1st Work. Scalable Resilient Infrastructures Distrib. Ledgers, Coloca. with ACM/IFIP/USENIX Middlew. 2017 Conf., no. 1, 2017, doi: 10.1145/3152824.3152830.

[11] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, J. Yellick, Hyperledger fabric: A distributed operating system for permissioned blockchains, 2018, https://doi.org/10.1145/3190508.3190538.

[12] M. Hill, What is scalability? ACM Sigarch Comput. Archit. News 18 (1990) 18–21.

[13] J. G. Shanahan and L. Dai, “Large scale distributed data science using apache spark,” Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., vol. 2015-Augus, pp. 2323–2324, 2015, doi: 10.1145/2783258.2789993.

[14] M. Krstić and L. Krstić, “Hyperledger frameworks with a special focus on Hyperledger Fabric,” Vojnoteh. Glas., vol. 68, no. 3, pp. 639–663, 2020, doi: 10.5937/vojtehg68-26206.

[15] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, “Performance modeling of PBFT consensus process for permissioned blockchain network (hyperledger fabric),” Proc. IEEE Symp. Reliab. Distrib. Syst., vol. 2017-Septe, pp. 253–255, 2017, doi: 10.1109/SRDS.2017.36.

[16] G. Chung et al., “Performance Tuning and Scaling Enterprise Blockchain Applications,” 2019, [Online]. Available: http://arxiv.org/abs/1912.11456.