Revue d'Intelligence Artificielle
Vol. 36, No. 4, August, 2022, pp. 581-587
Journal homepage: http://iieta.org/journals/ria

Performance Evaluation of E-Voting Based on Hyperledger Fabric Blockchain Platform

Shatha H. Saeed[1*], Suha M. Hadi[1], Ali H. Hamad[2]

[1] Commission for Computers and Informatics, Informatics Institute for Postgraduate Studies, Baghdad 10001, Iraq
[2] Department of Information and Communication Engineering, University of Baghdad, Baghdad 10071, Iraq

Corresponding Author Email: shatha.alkayal1975@gmail.com

https://doi.org/10.18280/ria.360410

Received: 24 June 2022
Accepted: 2 August 2022

Keywords:
blockchain, hyperledger fabric, hyperledger caliper, performance evaluation

ABSTRACT

Permissioned blockchain platforms have become more prevalent in a wide range of applications. These, such as hyperledger fabric platforms, are sensitive to latency and throughput. In this work, the E-voting case study adopts a hyperledger fabric platform where performance evaluation has been studied in terms of scalability, latency, throughput, CPU usage, and memory allocation. Three scenarios were performed with varying transaction rates, block size, and organizations. Another two scenarios were performed, first with varying block timeout and second, measuring the impact of CPUs and memory allocation on the proposed fabric's entities (peers, orderer, couchDB, chaincode, etc.). The result shows that an increase in block size will significantly affect metrics such as latency and throughput. Good results were obtained with high transaction send rates on large block size. Similarly, low performance is obtained using a small block size with increased send rates. Also, it was noticed that increasing the number of organizations will increase latency and decrease the throughput. Therefore, in applications with a large number of concurrent transactions, to maintain high throughput, block timeouts and block size should be large. On The other hand, the number of CPUs and amount of memory allocation would impact hyperledger fabric performance.

1. INTRODUCTION

A lot of data is generated and exchanged extremely fast in our daily lives. Therefore, platforms for data storage and exchange securely and reliably are necessary to manipulate this data. Usually, a third party entity controls and centralizes information transfer; for example a transaction in financial occurs between different entities, and a credit card service provider plays an important role in completing the transaction. Also, a fee is imposed by these providers for this trusted service [1]. Trust and intermediaries are the biggest problems facing data transfer; blockchain technology can overcome these issues where a decentralized and distributed ledger of encrypted transactions is used. Every single node within the blockchain network holds a copy of the database stored in blocks connected chronologically to form a chain of blocks. Blockchain technology is a distributed database structure that stores transactions where all nodes in the network must agree on the transactions and their order [2].

Each block in a blockchain is assigned a unique identity (hash) derived from the data included in that block and the hash of the block that preceded it. This enables the discovery of any modification to a block of data, as the modification will change the block's hash and the hashes of all subsequent blocks. The genesis block is the initial block in a blockchain and does not contain a hash of any prior blocks. All blocks may be verified by tracing them back to the genesis block. Once executed, a new transaction is broadcast to the whole blockchain network. Nodes known as block miners receive the transaction and confirm its authenticity by validating the signature. The authenticated transaction is then mined into blocks that are securely encrypted. For a block miner to generate a block, the consensus problem must be solved via distribution. Any miner that solves the consensus problem broadcasts the newly created block to the entire network. When a new block is received, miners who have not solved the consensus problem add it to their local chains because the transaction has been validated and the block includes the solution to the consensus problem. There are several consensus algorithms, however, the following are the most prevalent: proof-of-work, proof-of-stake, and practical-byzantine-fault-tolerance [3].

Two types of blockchain technology have been introduced which are permission and permissionless. This classification is determined by the node's capability to access or add a new block in the network. Although these platforms have some similar characteristics, their differences could affect requirements security that they fulfill. Most permissionless blockchain platform uses Proof of Work (PoW) for consensus which is computationally expensive and unsuitable for many applications that handle large of transaction. In a permissionless platform, permission is not required by a node to join a network, however, additional layers of privacy and control are needed for accessing and performing read/write ledger operations. Some examples of this type of platform are Bitcoin, and Ethereum [4, 5].

In permission blockchain platforms additional layers of security are provided, such as access control layer to handle permissions for specific operations by authorized nodes. Platforms such as Corda, Fabric, Multichain, and Quorum are permissioned blockchain. Hyperledger fabric is an open source that uses permissioned Distributed Ledger Technology

(DLT) designed for enterprise usage, which comprises a number of features that distinguish it from other popular distributed ledger platforms. In addition to having a highly modular and configurable architecture, fabric allows for innovation, versatility, and optimization across various industries including finance, banking, healthcare, human resources, and insurance. Smart contracts can be created on Fabric using general purpose programming languages including Java, Go, or Node.js, where fabric is the first platform that support these languages. Pluggable consensus protocols are one of the platform's most important differentiators, enabling it to be tailored to specific use cases and trust models, through its modular design, the platform can rely on well-established tools for Crash Fault Tolerant(CFT) or Byzantine Fault Tolerant(BFT) ordering. By leveraging consensus protocols over native cryptocurrencies, fabric can reduce mining costs or fuel smart contract execution without needing a native cryptocurrency [6]. Inspite of the many advantages of blockchain technology, it faces some challenges such as performance, privacy, and scalability. Performance is the main challenge in blockchain implementations, where the challenges included latency, throughput, bandwidth, size, versioning, wasted resources, hard forks, and usability. These challenges such as scalability, throughput, and latency have not been evaluated extensively. Evaluation is also required based on the number of peers, channels, organizations, nods, and transactions [7, 8].

In this paper, performance evaluation of using hyperledger fabric platform by measuring some metrics including average latency, throughput, and scalability modification such as block size, block timeout, number of endorser peers, and a number of organizations. Additionally, CPU percentage usage and memory allocation were evaluated to analyze the performance fabric's entities such as (chaincode, peers, orderer, etc). The E-voting system based hyperledger fabric platform is considered a case study. The hyperledger caliper benchmark has been used for the performance evaluation of hyperledger fabric with E-Voting. The rest of this paper is organized as follows: Section 2, introduces a list of related work in the performance evaluation of blockchain platforms. Section 3 discusses the background of hyperledger fabric. Section 4 proposed E-Voting system design with five scenarios conducted. Finally, section 5 describes the conclusion.

2. RELATED WORK

Recently performance evaluation of blockchain technology has the attention of many researchers due to its important role in developing new algorithms. Hang et al. [9] designed a fish farmer using a hyperledger fabric blockchain platform. Hyperledger caliper was used for measuring different evaluation metrics such as throughput and latency. Several experiments were done in this work to study the relationship between transaction send rate, latency, block size, and throughput. Tanwar et al. [10] proposed a hyperledger fabric for permission based electronic healthcare (HER) system, where different performance metrics have been used to evaluate the system such as throughput, latency, and round trip time (RTT). Hang and Kim [11] proposed a construction methodology based blockchain network to enhance the hyperledger fabric performance by observing different configurable network schemes. Kumar and Chand [12] proposed MedHypChain which is a healthcare medical system-based hyperledger fabric. The hyperledger caliper was used for performance analysis in three metrics: Execution time, latency time, and throughput. Lohachab et al. [13] proposed a secure peer to peer(P2P), cyber physical conceptual energy trading model with private and permissioned blockchain platforms. Polge et al. [14] provided a comprehensive and comparative study for different blockchain platforms (Fabric, Quorum, Ethereum, R3 Corda, and MultiChain) with performance, privacy, community activities, scalability, and adoption criteria. Swathi and Venkatesan [15] introduced permissioned blockchain to address scalability issues by incorporating distributed machine learning techniques. Díaz-Santiso and Fraga-Lamas [16] introduced a decentralized E-Voting system that provides a higher level of security, cost-efficiency, and transparency. Hyperledger fabric with smart contracts was used to cast votes. Transaction load and latency analysis of the proposed system were performed.

3. HYPERLEDGER FABRIC PLATFORM

The hyperledger project consists of a collection of open source subprojects such as Iroha, Sawtooth, Fabric, Indy, and Burro. Hyperledger fabric is a common open source permissioned blockchain [17]. Hyperledger fabric doesn't have any cryptocurrency like Bitcoin and Ethereum, in this platform access to the network is restricted to the network members only. A ledger in the fabric is comprised of two parts, a world state, and a blockchain. The world state is a database that stores the current values of ledger states. In addition to maintaining a versioned key-value store, fabric supports CouchDB and LevelDB as state databases [18]; In the present case study, CouchDB was used. The version number of each key is updated each time it is written. A blockchain records all successful and failed transactions (grouped into blocks) in a network. Hyperledger Fabric supports smart contract creation in general purpose languages (written in Go, Java, and NodeJs). A chaincode is a smart contract in the fabric where all functions that can be invoked by a transaction are defined. Chaincodes come with endorsement policies that apply to their linked smart contracts. An isolation mechanism called a channel can ensure the privacy of transactions between participants in a network. Every channel maintains its own ledger, ensuring the transaction and data are only available to member nodes in the channel. There are different types of entities in a fabric network, including peer nodes, orderer nodes, and clients that belong to different organizations. Each of these has an identity on the network provided by the Membership Service Provider (MSP). Certificate Authorities (CA) create identities by generating a public and private key pair that can be used to verify identity. Transactions are executed and peer nodes maintain ledgers. All transactions in the network are ordered by orderer nodes, which propose new blocks and seek consensus. Orders are collected in an ordering service, hyperledger fabric offers three ordering services (Solo, Kafka, and Raft) [19]. Every peer node is a committer by default, so it receives ordered state updates in the form of a block of transactions from the ordering service and maintains the ledger. When peer nodes receive a new block, they validate the transactions, commit the changes to the local copy of the ledger and append the block to the blockchain. Peer nodes can endorse transactions as well, thus being called endorsers. In order to gather endorsement proposals, the client proposes the transaction request simultaneously to many peers. The

transaction is then broadcast to the orderer to be included in a block and delivered to all peers for validation and commitment [20]. Invoke and Query are the two types of transactions, the invoke transaction runs the provided function with the specified arguments. It may entail reading and updating the state database, with success or failure as a result. The query transaction runs the supplied function, which returns the current state of the peer. As a result, only transactions that are invoked change the status of the distributed ledger. Execute, Order, and Validate are the three stages of a successful transaction. Figure 1 illustrates the transaction flow within hyperledger fabric.

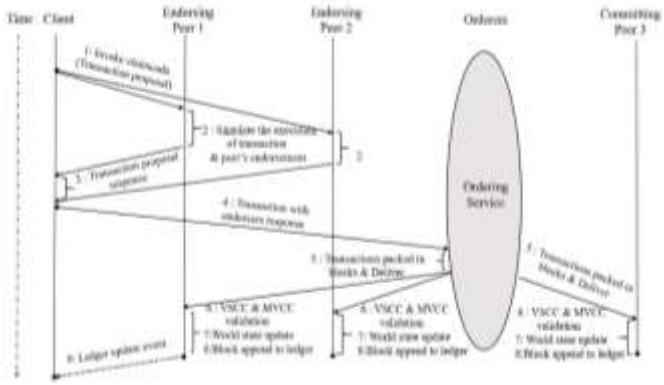


Figure 1. Transaction flow in hyperledger fabric

In the executing stage, the client application uses the fabric software development kit (SDK) to construct a transaction proposal to invoke the chaincode function, which reads and writes data to the ledger. Client credentials are used to sign the proposal, and one or more endorsing peers simultaneously receive it from the client. Then the peers perform a verification process for endorsement such as transaction format, duplication, issuer's signature, etc. In the order stage, the ordering service uses a consensus protocol to order transactions received from the client. A transaction block is generated when one of three conditions is met: block timeout, block size, and block maxbytes. The final stage is the validation where every peer validates the orderer's signature on the block, where the process is as follows: first peers validate the orderer's signature on the block, second signatures are decoded and finally evaluate endorsement policy using validation system chaincode (VSCC) that check if a sufficient number of related endorsing peer signature, the peer then checks the key version using Multi Version Concurrency Control(MVCC). If VSCC and MVCC validation checks are passed, the write sets to the world state otherwise the validation stage is failed. Peers can notify subscribed clients about the commit event [21].

4. PROPOSED E-VOTING SYSTEM DESIGN

Performance evaluation of hyperledger fabric using hyperledger caliper under Linux operating system has been proposed with E-voting case study. The proposed system consists of client nodes (voting stations) that submit voting as transactions to the client application (E-Voting APP) server, fabric software application development kit(SDK) allows a client application to interact and connect to all configured members of the proposed fabric network. Peers' node (committer node and endorser node) arranged into an organization, orderer service node, and channel represent fabric network members. Each organization have, a certificate authority (CA) to generate the certificates that represent identities (key pairs), and a membership service provider (MSP) contains a list of permissions identities. Figure 2 illustrated the architecture of the proposed system design.

Once the election's voting period has been set to start by the election commission authority, the registered voters could only cast their vote during this period. Therefore, it extremely leads to an increased number of concurrent voting transactions.

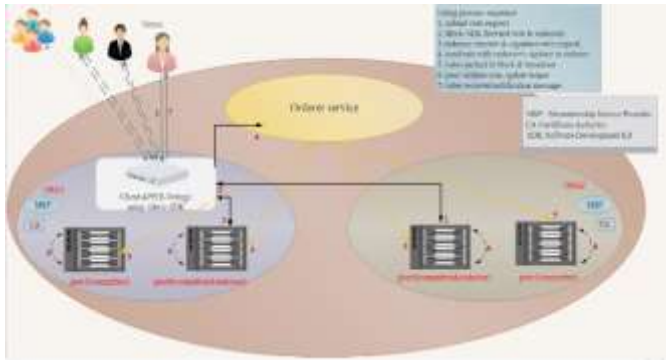


Figure 2. The architecture of the proposed E-voting system is based on the hyperledger fabric network

Three different scenarios have been proposed: one, two, and three organizations (two peers per organization and one endorser peer in each organization) with two metrics to monitor; throughput and average latency. Throughput is the rate at which transactions are successfully processed when it is included in a block and committed to the ledger as part of the blockchain in transactions per second. Throughput was given in Eq. (1), where *TRT* is transaction throughput, *tct* is total committed transaction, *tti* is total time.

$$TRT = \frac{tct}{tti} \tag{1}$$

Latency could be calculated by Eq. (2), where *TRL* is transaction latency, *cti* is transaction committing timestamp, and *sti* is transaction submission timestamp.

$$TRL = cti - sti \tag{2}$$

The committing timestamp is the time when a transaction is committed to the ledger. While the average latency is the average of the total transaction latency. Average latency was given in Eq. (3), where *ATL* is average transaction latency, *ttl* is total transactions latency, and *tct* is the total committed transaction.

$$ATL = \frac{ttl}{tct} \tag{3}$$

To measure the different impacts on the performance metrics, five incoming transaction rates ranging from (50 tps, 100 tps, 150 tps, 200 tps, and 250 tps), have been considered. The transactions are sent in parallel and generated by an application were multiple client nodes issue transactions simultaneously. The required load is generated by all transactions. For every transaction, each node will issue 1000 transactions. The network will then validate the transactions and append new blocks to the blockchain. Then, the fabric

network is scaled to multiple numbers of endorsement peers and organizations. Additionally, the varying block sizes specifically (10, 50, 100, 150, and 200) transactions per block have been tested for each scenario. Algorithm .1 shows the sequence of voting transactions.

Algorithm 1: *Voting transaction*

```
1   Input:  VoteID , VoterID, candidateID
    Start:
2   Voter registration process
3   If  Start (voting time)  {
4       Voter submits transaction
5       Transaction request forward to endorsing peers
6       Start  endorsing peers  simulate execution phase
7       If  !has-voted(VoteID)  AND  vote-time!= end  {
8   .           endorser's signature=True
9       else
10              transaction= False
11          }
12          Forward vote with endorser's signature to
            orderer service
13          Start  ordering phase{
14          voting transaction packed in block
15          Broadcast block to all peers
16                  }
17      Start  peer validate phase  {
18      If  (VSCC) and (MVCC)  {
19          Vote transaction commit in the world state
20      else
21          Committing to the world state failed
22        }
23  }
24  Send a notification message to voter
25  End.
```

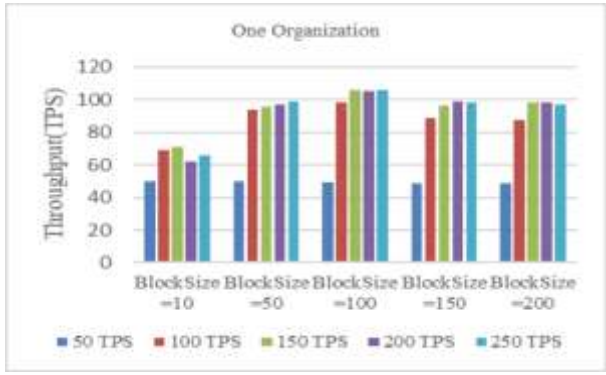
Three scenarios have been proposed: one, two, and three organizations, for each scenario (two peers per organization and one endorser peer in each organization). Each time block size is increased, five rounds (round is an array of objects, each describing the setting of a round) are run. Every round set with 1000 transactions on a fixed send rate (50 tps, 100 tps, 150 tps, 200 tps, and 250 tps). In these scenarios (2sec) block timeout is specified. Performance evaluation could help to measure the impact of scaling the organization and peers when varying block size, Table 1 illustrate basic configuration parameters.

Table 1. Basic configuration parameters

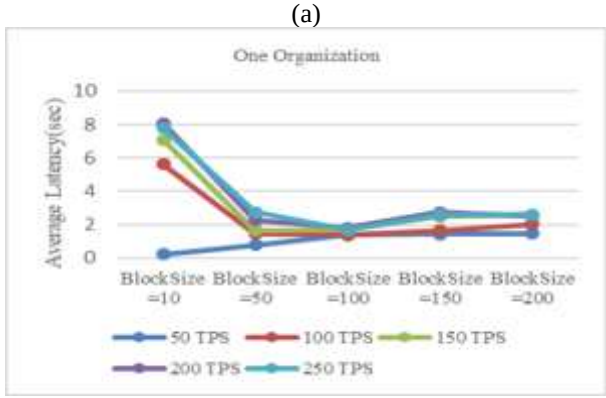
Parameter	Configuration
Round	5 rounds (each round set with one specific send rate)
Transactions	1000 transactions per round
Transaction mode	Read, Write (Vote)
Send Rate	(50,100,150,200,250) tps
Block Size	(10.50,100,150,200) transactions per block
Block TimeOut	2 sec

i. One organization scenario

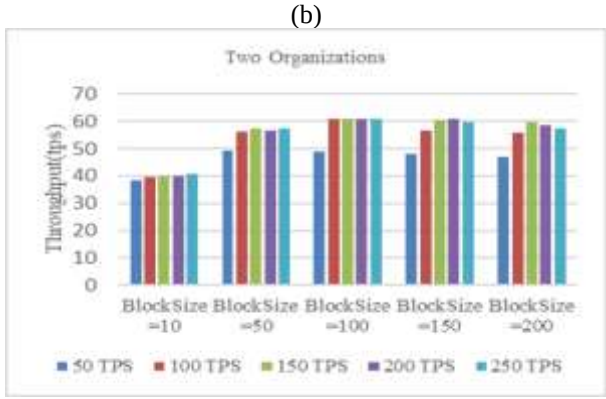
In this scenario, performance evaluation of varying block size and transaction send rate has been tested. Figure 3(a) shows the transaction throughput, where it is increased when increasing in block size up to 100 transactions per block, the highest throughput obtained was about (106 tps). While the block size increased (150 and 200) transactions per block, it is noted that the throughput slightly starts to decrease. Figure 3(b) shows that the average latency, where it is decreases when increasing in the block size up to 100 transactions per block. When the block size increases to (150 and 200) transactions per block, average latency starts to increase slightly. This implies that this specific test environment when increasing block size higher than 100 transactions per block does not have a significant impact on performance. Furthermore, better average latency and performance were noted for small send rates, like 50 transactions per second, when using smaller block sizes, such as 10 transactions per block, so increasing block sizes did not greatly impact performance, but it was still noticeable.



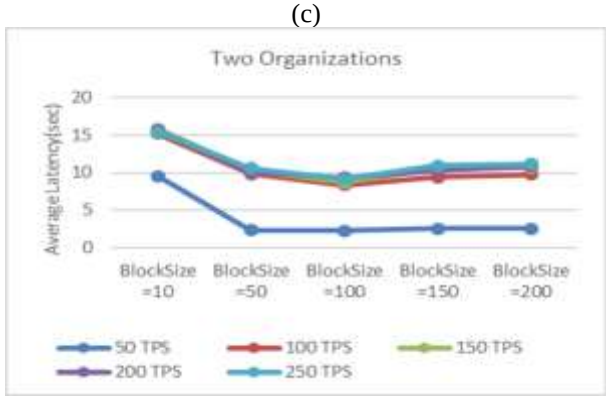
(a)



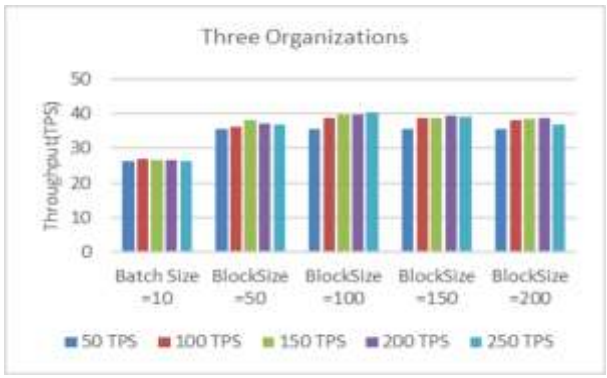
(b)



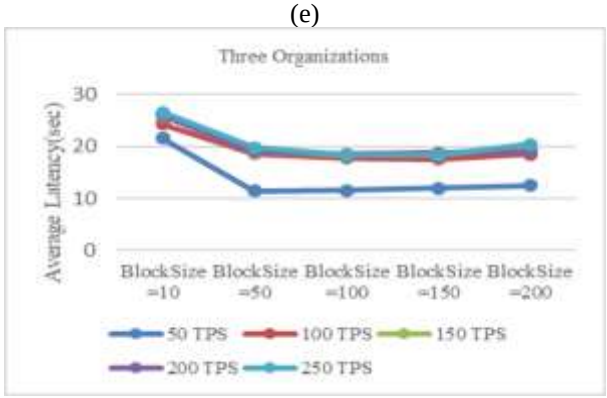
(c)



(d)



(e)



(f)

Figure 3. The results of varying block size experiments

ii. Two organizations scenario

According to this scenario, a study of the performance evaluation of varying block size and transaction send rate has been tested. Figure 3(c) shows that the transaction throughput increased when increasing in block size up to 100 transactions per block, where (61 tps) was the highest throughput obtained. While when the block size increased to (150 and 200) transactions per block, the throughput is slightly decreased. Figure 3(d) shows that the average latency decreases when increasing in the block size up to 100 transactions per block, the growth in latency was slightly higher when block size increases to (150 and 200). The results indicate that increasing block size to higher than 100 transactions per block does not have a significant impact on performance. Better performance and latency were observed for small send rates, like 50 tps, when using a smaller block size like 50 transactions per block, so increasing the block size did not affect performance that much, even so, it is still noticeable.

iii. Three organizations scenario

Based on this scenario, performance evaluation of varying block size and transaction send rate has been tested. Figure 3(e) shows that increased transaction throughput when increasing in block size up to 100 transactions per block, the highest throughput obtained was (40.3 tps). During the block size increased by (150 and 200) transactions per block, the throughput of each block size started to slightly decreased. Figure 3(f) shows that the transaction average latency decreases when increasing in block size up to 100 transactions per block, while average latency growth slightly increases by increasing block size to (150 and 200) transactions per block. Another observation was found, for a small send rate like (50 tps) was better throughput and latency with a smaller block size like 50 transactions per block, which means increasing the block size had a small effect on performance, but the difference is quite small.

iv. Two organizations with varying block timeout scenario

Another scenario has been conducted when varying block timeout (5sec, 2sec, 0.5ses, and 0.25sec, 0.1sec) to evaluate the performance of the proposed fabric network consisting of two organizations and two peers (one endorser peer in each organization) per organization and fixed block size to 100 transactions per block. The impact of varying block timeout on performance over different transaction send rate were evaluated. Table 2. shows the network configuration parameters of the caliper used to perform this scenario. Two organizations with two peers (one endorser in each organization) per organization configured a network scenario with a fixed block size (100 transactions per block) was selected for the following reasons:

- Despite one organization with two peers (one endorser in each organization) scenario obtained the best performance as discussed in scenario (i), but it's not suitable due to have a single endorser peer, so any fail on it leads to rejecting all transaction proposal and stop the transaction workflow.
- Three organization with two peers (one endorser in each organization) scenario among other scenario obtained the worst performance as discussed in scenario (iii), despite it has three endorser peers distributed over three organizations and any endorser peer's failure does not effect on transactions workflow, but it's latency increases when the number of parallel transactions increases, this because when increase endorsers, the client waits for endorsers response before sending transactions to the orderer.
- Block size (100 transactions per block) was fixed for this scenario because it obtained the highest throughput and lowest latency among all the previous three scenarios.

Table 2. Network configuration with varied block timeout measurement

Parameter	Configuration
Round	5 rounds
Network size	2Org 2Peer(one each organization's peer as an endorser peer)
Transactions	1000 transactions per round
Transaction mode	Read, Write (Vote)
Send Rate	(50,100,150,200,250) tps
Block size	(100) transactions per block
Block Timeout	5sec, 2sec, 0.5sec, 0.25sec, 0.1sec

Figure 4(a) shows that the throughput increases when increasing block timeout. For 5sec block timeout the highest throughput obtained was (97 tps). While when the block timeout decreased to (2 sec, 0.5 sec, 0.25 sec, and 0.1 sec), the throughput of each case is slightly decreased. Figure 4(b) shows that the re transaction average latency decreases when increase the block timeout, the highest latency dose not reached (2 sec) when block timeout was (5 sec), while the highest latency reached was (10 sec, 11 sec, 12 sec, 13 sec) when decreased block timeout to (2 sec, 0.5sec, 0.25 sec, and 0.1 sec) respectively. A decreasing in block timeout implies cutting the block before it reached a specific capacity. These results indicate that the increasing of block timeout has the better performance when increasing block sizes.

v. The impact of CPU usage and memory allocation on performance

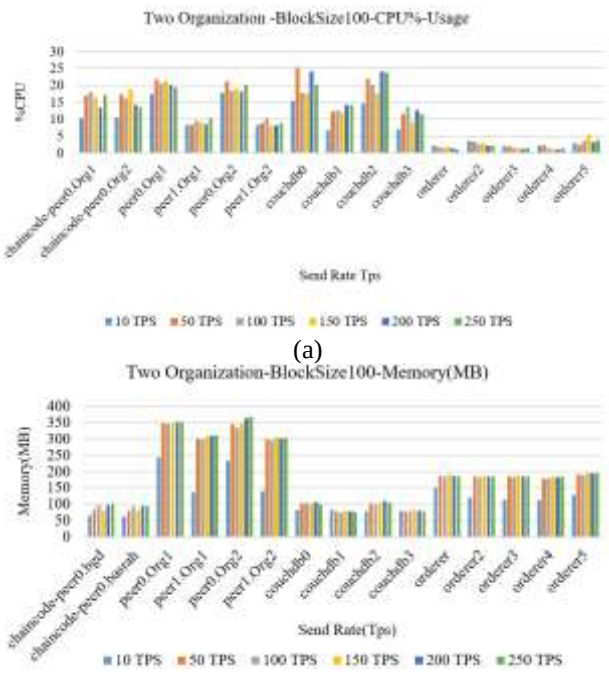
In this scenario Two organizations with two peers (one endorser in each organization) per organization configured a network scenario with a fixed block size (100 transactions per

block) was selected for the same reasons discuses in scenario (iv). Also block timeout was fixed to (2 Sec). To evaluate the impact of CPU usage and memory allocation on proposed network performance. Figure 5(a) shows that the CPU usage will be high by couchDB0 and couchDb2 followed by peer0.Org, peer0.Org2, chaincode.peer0.Org1,and chaincode.peer0.Org2. This indicate that execution and validation stage consume high CPU than other stage. Figure 5 (b) shows the memory allocation will be high in peer0.Org1, and peer0.Org2(which are endorsing and committing peers) followed by peer1.Org1, and peer1.Org2 which are committing peers).

(a)

(b)

Figure 4. The results of the varying block timeout experiment



(a)

(b)

Figure 5. The CPU usage and memory allocation by Hyperledger Fabric's entities

5. CONCLUSIONS

This paper discusses the effect of the E-Voting workload on the performance of the hyperledger fabric blockchain platform in terms of latency, scalability, and throughput. Several scenarios were performed by varying transaction send rates (tps), block size, block timeout, and the number of organizations. In summary, the blockchain network depends on network design, smart contract complexity/operations, and hardware configuration. So, in more detail, the following conclusions were reached:

(1) Increase block size significantly affects the performance of the blockchain network, particularly in latency and throughput.

(2) The type of transactions (i.e. invoke or query transaction) affects the network latency due to complexity and the number of operations involved.

(3) Latency increases with the increase in number of organizations.

(4) Good results were obtained with high transaction send rates on large block size. Similarly, low performance is obtained when using a small block size with increasing in send rates.

(5) At low transaction arrival rates, a low block size is preferable, so that the ordering service does not have to wait for a long time until enough transactions have arrived for a block to be created. However, as the transaction arrival rate increases, this delay becomes less significant.

(6) In applications with a large number of concurrent transactions, block timeouts and size should be large to maintain high throughput.

(7) The number of CPUs and amount of memory significantly affect hyperledger fabric performance.

ACKNOWLEDGMENT

This work is supported by the Informatics Institute for Postgraduate Studies, Commission for Computers and Informatics.

REFERENCES

[1] Dabbagh, M., Choo, K.K.R., Beheshti, A., Tahir, M., Safa, N.S. (2021). A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities. Computers & Security, 100: 102078. http://dx.doi.org/10.1016/j.cose.2020.102078

[2] Dinh, T.T.A., Liu, R., Zhang, M., Chen, G., Ooi, B.C., Wang, J. (2018). Untangling blockchain: A data processing view of blockchain systems. IEEE Transactions on Knowledge and Data Engineering, 30(7): 1366-1385. http://dx.doi.org/10.1109/TKDE.2017.2781227

[3] Androulaki, E., Barger, A., Bortnikov, V., et al. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth Eurosys Conference, pp. 1-15. https://doi.org/10.1145/3190508.3190538

[4] Nasir, Q., Qasse, I.A., Abu Talib, M., Nassif, A.B. (2018). Performance analysis of Hyperledger fabric platforms. Security and Communication Networks, 2018: 3976093.

http://dx.doi.org/10.1155/2018/3976093
[5] Khan, D., Jung, L.T., Hashmani, M.A., Cheong, M.K. (2022). Empirical performance analysis of Hyperledger LTS for small and medium enterprises. Sensors, 22(3): 915. https://doi.org/10.3390/s22030915
[6] Berendea, N., Mercier, H., Onica, E., Riviere, E. (2020). Fair and efficient gossip in Hyperledger fabric. In 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), pp. 190-200. http://dx.doi.org/10.1109/ICDCS47774.2020.00027
[7] Kuzlu, M., Pipattanasomporn, M., Gurses, L., Rahman, S. (2019). Performance analysis of a Hyperledger fabric blockchain framework: Throughput, latency and scalability. In 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, pp. 536-540. http://dx.doi.org/10.1109/Blockchain.2019.00003
[8] Hyperledger Caliper Architecture. Available online https://hyperledger.github.io/caliper/v0.3.2/architecture/, accessed on 27 May 2022.
[9] Hang, L., Ullah, I., Kim, D.H. (2020). A secure fish farm platform based on blockchain for agriculture data integrity. Computers and Electronics in Agriculture, 170: 105251. http://dx.doi.org/10.1016/j.compag.2020.105251
[10] Tanwar, S., Parekh, K., Evans, R. (2020). Blockchain-based electronic healthcare record system for healthcare 4.0 applications. Journal of Information Security and Applications, 50: 102407. http://dx.doi.org/10.1016/j.jisa.2019.102407
[11] Hang, L., Kim, D.H. (2021). Optimal blockchain network construction methodology based on analysis of configurable components for enhancing hyperledger fabric performance. Blockchain: Research and Applications, 2(1): 100009. http://dx.doi.org/10.1016/j.bcra.2021.100009
[12] Kumar, M., Chand, S. (2021). MedHypChain: A patient-centered interoperability Hyperledger-based medical healthcare system: Regulation in COVID-19 pandemic. Journal of Network and Computer Applications, 179: 102975. http://dx.doi.org/10.1016/j.jnca.2021.102975
[13] Lohachab, A., Garg, S., Kang, B.H., Amin, M.B. (2021). Performance evaluation of Hyperledger Fabric-enabled framework for pervasive peer-to-peer energy trading in smart Cyber–Physical Systems. Future Generation Computer Systems, 118: 392-416. http://dx.doi.org/10.1016/j.future.2021.01.023
[14] Polge, J., Robert, J., Le Traon, Y. (2021). Permissioned blockchain frameworks in the industry: A comparison. ICT Express, 7(2): 229-233. http://dx.doi.org/10.1016/j.icte.2020.09.002
[15] Swathi, P., Venkatesan, M. (2021). Scalability improvement and analysis of permissioned-blockchain. ICT Express, 7(3): 283-289. http://dx.doi.org/10.1016/j.icte.2021.08.015
[16] Díaz-Santiso, J., Fraga-Lamas, P. (2021). E-Voting system using Hyperledger fabric blockchain and smart contracts. Engineering Proceedings, 7(1): 11. https://doi.org/10.3390/engproc2021007011
[17] The Linux Foundation Project. Available online: https://www.hyperledger.org, accessed on 27 May 2022.
[18] Chacko, J.A., Mayer, R., Jacobsen, H.A. (2021). Why do my blockchain transactions fail? A study of Hyperledger fabric. In Proceedings of the 2021 International Conference on Management of Data, pp. 221-234. http://dx.doi.org/10.1145/3448016.3452823
[19] Shalaby, S., Abdellatif, A.A., Al-Ali, A., Mohamed, A., Erbad, A., Guizani, M. (2020). Performance evaluation of Hyperledger fabric. In 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), pp. 608-613. http://dx.doi.org/10.1109/ICIoT48696.2020.9089614
[20] Thakkar, P., Nathan, S., Viswanathan, B. (2018). Performance benchmarking and optimizing hyperledger fabric blockchain platform. In 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 264-276. http://dx.doi.org/10.1109/MASCOTS.2018.00034
[21] Sukhwani, H., Wang, N., Trivedi, K.S., Rindos, A. (2018). Performance modeling of Hyperledger fabric (permissioned blockchain network). In 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), pp. 1-8. http://dx.doi.org/10.1109/NCA.2018.8548070