2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS) | 978-1-6654-9172-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/DAPPS55202.2022.00016

Performance Analysis of Hyperledger Besu in Private Blockchain

Caixiang Fan
University of Alberta
Edmonton, Canada
caixiang@ualberta.ca

Changyuan Lin
York University
Toronto, Canada
chlin@yorku.ca

Hamzeh Khazaei
York University
Toronto, Canada
hkh@yorku.ca

Petr Musilek
University of Alberta
Edmonton, Canada
Petr.Musilek@ualberta.ca

***Abstract*—In this work, we present a set of comprehensive experimental studies on Hyperledger Besu in private blockchain. We aim to exhibit its performance characteristics in terms of transaction throughput, latency, resource utilization, and scalability, from the application perspective by adding a load balancer middleware. We have carefully designed a set of comparative experiments and judiciously selected typical parameters, including transaction send rate, network size, node flavor, load balancing, consensus, and block time. In particular, three proof of authority consensus algorithms, Clique, IBFT 2.0, and QBFT, are investigated. Through extensive experimental evaluations using the Hyperledger Caliper benchmark tool, we analyze how these parameters impact the performance of a private Besu blockchain. Our studies reveal several interesting findings: 1) Blockchain parameters, e.g., block time and block size, are the most significant factors in determining Besu performance; 2) The performance of Besu is bottlenecked by transaction execution and blockchain state updates, which are determined by parameters such as node computation power, transaction complexity, and load balancing; 3) A Besu network with QBFT consensus can scale up to 14 validators without noticeable performance loss. Our findings shed some light on further performance improvement of Hyperledger Besu. The identified bottlenecks and root cause analysis provide insightful suggestions for blockchain practitioners to build performant enterprise applications.**

***Index Terms*—Performance Analysis, Scalability, Hyperledger Besu, Private Blockchain, Load Balance.**

I. Introduction

Beyond cryptocurrency, blockchain technology has been adopted by many enterprise applications, such as supply chain [1]–[3], healthcare [4], [5], and peer-to-peer transactions [6]–[8], to overcome security and privacy issues in traditional centralized solutions and build trust between participants by removing the trusted third party. Decentralization, immutability, and enhanced security make blockchain a promising platform for enterprises to store and share data among participants in a network. From the perspective of network accessibility, blockchain can be divided into two main categories: public and private. Unlike a public blockchain, where the participant does not need permission to join the network, a private blockchain usually requires permission and assumes a certain degree of trust among entities. Due to the benefits of verified participants, enterprises can build decentralized applications on a private blockchain with an efficient consensus to achieve immediate finality, high performance, and improved privacy.

Recently, Ethereum has become one of the mainstream blockchain platforms for developing enterprise applications. An Ethereum network consists of nodes, called clients, running software that executes transactions, verifies, and creates blocks. All nodes in the network maintain a state machine, namely the Ethereum Virtual Machine (EVM), which is responsible for executing transactions and smart contracts to compute the network's state from block to block. A smart contract is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain and automatically enforces the pre-defined rules to execute as programmed.

There exist many Ethereum clients implemented in various programming languages, such as Go, JavaScript, Python, and Java, by following a formal specification. Although enterprises can build a private blockchain by initiating a genesis file using regular clients, such as Geth, Nethermind, and OpenEthereum, this network can hardly meet the specific enterprise needs. To address this problem, Enterprise Ethereum emerges to extend public Ethereum by providing enterprise specifications, including the capability to enforce membership (permissioning), provide high performance, and perform private transactions, which allow only participants of those transactions to access the metadata or payload. As one of the most popular Enterprise Ethereum clients, Hyperledger Besu has recently gained much attention in constructing enterprise decentralized applications [9], [10] due to its versatility, improved privacy, and high performance.

However, the performance and scalability of Besu, e.g., the impact of system configurations and chain parameters on the performance metrics of throughput and latency, has not been thoroughly studied. In this paper, we propose a load balance-based performance evaluation architecture, as shown in Figure 1, and analyze the performance of Hyperledger Besu from the application perspective in private blockchains, where proof of authority (PoA) consensus is preferred over proof of work (PoW) and proof of stake (PoS). In particular, we analyze three PoA consensus algorithms supported by Besu, namely Clique, Istanbul Byzantine Fault Tolerance (IBFT) 2.0, and QBFT, through extensive benchmark experiments and log analysis. The main contributions of this work are as follows:

- We present an in-depth experimental evaluation of Hyperledger Besu for its PoA consensus algorithms, showing its

DOI 10.1109/DAPPS55202.2022.00016

performance characteristics under various configurations.
- Through extensive benchmarks, we identify critical parameters that impact the performance and scalability of Besu and analyze to what extent the influence is.
- We analyze the root causes of the identified bottlenecks through log analysis.

The remainder of this paper is organized as follows: In Section II, we elaborate on the basics of Hyperledger Besu. In Section III, we present the experimental evaluation of the proposed benchmark framework. Section IV outlines the experimental results achieved. In Section V, we analyze the root causes of how the identified parameters impact the performance of Besu. In Section VI, we review the latest related work on private blockchain performance analysis. Section VII concludes the paper and discusses potential future works.

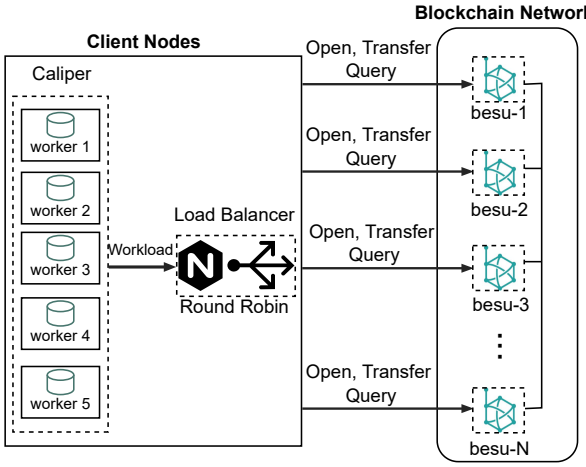

Fig. 1: Hyperledger Besu performance evaluation architecture.

II. Hyperledger Besu Overview

Hyperledger Besu is an Ethereum client implementing the Enterprise Ethereum specifications, which are maintained by the Enterprise Ethereum Alliance (EEA), a membership of blockchain and incumbent businesses from around the world. It was initially known as Pegasys Pantheon, then joined Hyperledger Foundation and was renamed Hyperledger Besu in 2019. Besu becomes popular because of its versatility. Its latest version (v22.4, May 2022) supports three types of consensus protocols: PoW (Ethash), PoS, and PoA (Clique, IBFT 2.0, QBFT). A summary of all consensus protocols that Besu supports is listed in Table I. We will focus on proof of authority consensus protocols used in private networks.

A. Clique

Clique is a PoA consensus protocol presented in the Ethereum Improvement Proposal (EIP) 225 [11]. In this protocol, only a fixed set of authorized nodes, called signers, can collect and execute the transactions from the transaction pool into a block and update the world state. It maintains a signer list through a voting mechanism, in which existing signers propose and vote to add or remove signers. The process of creating or mining a block is called *sealing*. Signers create the next block in a round-robin fashion at a fixed interval determined by *blockperiodseconds*. The order will be determined by the block number and the signer count. In order to prevent a malicious signer from sealing fraudulent transactions, a signer is only allowed to seal a block every $\lfloor N/2 \rfloor + 1$ blocks. Therefore, at any point in time, there are only $N - \lfloor (N/2) + 1 \rfloor$ signers who are allowed to seal a block. In Clique, forks occasionally happen due to concurrent blocks. In addition, if a designated signer cannot seal a block in time, any other signers may propose a block after waiting for the block period, which is called out-of-order sealing. Out-of-order sealing can happen quite frequently in case of short block period times and significant network delays between nodes, which will negatively impact the network performance.

Therefore, Clique does not have immediate finality. Implementations using Clique must be aware of forks and chain reorganizations occurring. Besu implements Clique, which can be used in a private network or test networks such as Rinkeby and Goerli. But, it is not recommended for production. Other Ethereum clients implementing Clique consensus include Geth and Quorum.

TABLE I: A comparison of Besu's consensus protocols.

Property	Clique	IBFT2	QBFT	Ethash	PoS
Type	PoA	PoA	PoA	PoW	PoS
Finality	No	Yes	Yes	No	No
Liveness	1/2	1/3	1/3	1/2	1/2
Throughput	High	High	High	Low	Medium
Quorum	1/2	2/3	2/3	1/2	1/2
BFT	No	Yes	Yes	Yes	Yes
Usecases	Private, Rinkeby, Goerli	Private	Private	Public	Public

B. IBFT 2.0

Istanbul Byzantine Fault Tolerance (IBFT) is a variant of PBFT [12] suitable for blockchain networks. It was first proposed informally in EIP 650 [13] and was implemented in Geth by AMIS Technologies in 2017, and soon in Quorum. As a leader-based (or voting-based) consensus, IBFT is deterministic and can tolerate f faulty participants out of n, where $n \leq 3f + 1$. It inherits from the original PBFT by achieving consensus in three phases, *pre-prepare*, *prepare* and *commit*, and has $O(n^2)$ total communication complexity. Before each round, the selected proposer will propose a new block proposal and broadcast it along with the *pre-prepare* message. Upon receiving the *pre-prepare* message, other validators enter the state of *pre-prepared* and then broadcast *prepare* message. When receiving $2f + 1$ *prepare* messages, the validator enters *prepared* state and then broadcasts *commit* message. Finally, validators wait for $2f + 1$ *commit* messages to enter the *committed* state and then attach the proposed block to the chain.

However, Saltini and Hyland-Wood [14] from ConsenSys analyzed the correctness of the initial version of IBFT and

pointed out that it does not guarantee Byzantine-fault-tolerant consistency and liveness when operating in an eventually synchronous network. They further resolved the liveness and finality issues in IBFT 1.0 by proposing a new version, called IBFT 2.0 [15]. Specifically, they modified the number of nodes for reaching consensus from $2f+1$ to super-majority, and also improved the round change by removing the block locking mechanism. Besu quickly deprecated IBFT 1.0 and replaced it with IBFT 2.0 after the new version was developed in 2020. So, unless otherwise specified, IBFT or IBFT2 refers to IBFT 2.0 throughout this paper.

C. QBFT

To further resolve the safety and liveness issues [16] in IBFT where the whole system can get stuck because of two honest nodes locking on different blocks, Quarum blockchain proposed and developed a variant of IBFT, called QBFT [17]. Like IBFT and PBFT, QBFT employs the famous three-phase commit scheme: *pre-prepare*, *prepare* and *commit*. For each round, a block proposal is created and broadcast with a pre-prepare message to other validators. Other validators broadcast a prepare message upon receiving the pre-prepare message. Then, on receiving the prepare message, a validator sends a commit message. Finally, a new block is inserted into the chain by the proposer after $2N/3$ commit messages. If a consensus is not achieved among all validators before the pre-defined time, a round change will happen with all clock time being reset. To ensure safety across round changes, QBFT employs a justification mechanism of proposed values that is critical in achieving quadratic communication complexity [17]. Besu fully implemented QBFT in the release of v22.1.1 and recommended it as the enterprise-grade consensus for production in place of IBFT 2.0.

III. Experimental Evaluation

In this section, we introduce our evaluation methodology, experimental setup, and configurations. We extensively tested varying configurations, including network size, node flavor, load balancing, consensus algorithm, and block period seconds. In particular, we used the control variate method to explore the impact of each parameter on the performance of Besu. All tested parameters are summarized in Table II.

TABLE II: Tested parameters in the experiments.

Test Parameters	Varying	Fixed
Send rate	$100 \rightarrow 10,000$ with step=100	8-LB-2C7.5G-QBFT-1S
Network size(N)	$4 \rightarrow 36$ with step=2	⋆-LB-2C7.5G-QBFT-1S
Load balance(LB)	LB, NLB	8-⋆-2C7.5G-QBFT-1S
Node flavor(F)	1C7.5G, 2C7.5G, 2C15G, 4C15G, 4C30G, 8C30G, 16C60G	8-LB-⋆-QBFT-1S
Consensus(C)	Clique, IBFT2, QBFT	8-LB-2C7.5G-⋆-1S
Block time(BT)	1S, 2S, 3S, 4S, 5S	8-LB-2C7.5G-QBFT-⋆

Here, we organize our configurations into the formation of "N-LB-F-C-BT". If a parameter is missing, it indicates that this parameter is under test with varying values. By default, the transaction send rate (i.e., the designated total number of transaction requests sent from all workers per second, measured in req/s) varies from 100 to 1,000 with a step of 100, which is configured by Caliper [18] at each test. According to the minimum system requirements (a VM with 6GB RAM and 20GB HDD) for a private Besu network, we define the configurations in the first line in Table II as the baseline, which has 8 nodes, each with 2vCPUs, 7.5GB RAM, and 36GB SSD. Other flavors have at least 36GB of SSD storage. All artifacts (i.e., source codes, configuration files, logs, and experimental results) of this article are publicly available in our reproducible repository[1].

A. Experimental Setup

We performed our experiments on an OpenStack-based cloud with varying configurations shown in Table II. For each configuration, for example, "8-LB-2C7.5G-QBFT-1S", we deployed a private Hyperledger Besu network with the latest version v22.4 at the time of writing on an OpenStack-based cloud. Each Besu node runs in a Docker (20.10.14) container on a virtual machine with Ubuntu 20.04 OS installed. The cross-node communication uses host networking, which has a bandwidth of 1,000Mb/s and network delay of less than 1ms. We also ran a network time protocol (NTP) service on each node to synchronize its time with a configured server, ensuring the time offset between any two nodes is less than 1ms. Besu DEBUG logs were collected through Docker logging for further analysis.

To avoid the limitation of gas consumption in smart contract deployment and transaction execution, we configured all Besu blockchain networks to be gas-free by setting "gasLimit" to the maximum "0x1fffffffffffff", "contractSizeLimit" to the maximum 2147483647, and the node startup option "min-gas-price" to 0. Also, we set the difficulty to the minimum "0x1" and the transaction pool queue size to the default 4096. For simplicity, all nodes were configured as validators (IBFT and QBFT) or signers (Clique).

Then, we used the benchmark tool Hyperledger Caliper v0.4.2 with Ethereum SDK v1.4 to connect and test the deployed Besu blockchain networks. Three types of performance metrics are collected: transaction/query throughput, transaction/query latency, and node resource utilization (CPU and memory). In each test, Caliper was running in a container on a client VM with rich resources (e.g., 16vCPU and 60GB RAM). Docker monitor was configured to collect the resource utilization of all the remote Besu node containers. The results were stored in performance reports.

B. Workloads

To comprehensively test the key parameters listed in Table II, we leverage the popular simple contract [19] with three types of transactions (i.e., open, query and transfer) to generate workloads. Open transaction performs one single

[1]https://github.com/pacslab/besu

write operation to the blockchain by assigning a value to an account through mapping; query performs a single read operation; transfer performs two write operations by adding to receiver account and subtracting from sender account. For each type of transaction, we set up the workload to 1,000 transactions, which will be sent at a fixed rate in a single test. The transaction send rate changes from 100 to 1,000 in 10 tests for all configurations except for the baseline heavy load test, where the transaction rate ranges from 100 to 10,000. From Table II, we have 30 different configurations, and for a single configuration under a specific transaction send rate, we run 25 replicas to generate 25 performance reports. Therefore, over 22,500,000 transactions have been generated and sent to the deployed Besu networks in our experiments.

C. Performance Metrics

In this section, we introduce four performance metrics, i.e., throughput, latency, resource consumption, and scalability, which are evaluated in this paper. The first three are the basic metrics supported by Caliper, while scalability is an extended metric indicated by the basic metrics in varying sizes of Besu networks. Here, we define all the evaluated performance metrics as follows:

- **Throughput** is the number of successfully committed transactions or query operations per second. The transaction throughput is expressed as transactions per second (TPS) at a network size. Query throughput is expressed as transactions per second (TPS) from a single node.
- **Latency** is the amount of time taken for a transaction's effect to be usable across the network. The measurement includes the propagation time and the consensus time. Query latency is the time between when the read request is submitted and when the reply is received.
- **Resource consumption** is the CPU and memory (RAM) utilization in each node container.
- **Scalability** is the ability to support increasing network participants or computation resources of nodes. This measurement is indicated by the throughput and latency when the network or node size increases.

D. Load Balancer

The load balancer is a crucial component for distributed systems deployed in the production environment. In our setting, we add a middleware layer between Caliper and the Besu network, where an NGINX-based load balancer leverages the round-robin algorithm to distribute all incoming Remote Procedure Call (RPC) transaction requests evenly across the Besu nodes. The load balancer can optimize resource utilization of upstream nodes, pass requests to other nodes if a node fails or has high tail latency, limit the number of connections under heavy load, and place the request in a queue for further processing. Therefore, the load balancer can generally increase the throughput, reduce the latency, and improve the availability and fault tolerance of the Besu network. In addition, it makes the performance metrics derived from our experiments more aligned with those in the production environment.

IV. Experimental Results

In this section, we will go through our experimental results and discuss them in detail.

A. Baseline

To explore when the network with baseline configuration becomes saturated, we tested Besu by varying the transaction send rate from 100 req/s up to 10,000 req/s. Figure 2 shows the performance of Besu under heavy load. We observe that the client could generate up to 1,485 open, 1,570 transfer, and 3,028 query transactions per second, respectively. But, the Besu network could only deal with up to 442 open, 414 transfer, and 3,007 query transactions per second, respectively. The query throughput is limited by the actual transaction arrival rate, which is upper-bounded by the maximum query request rate that the Caliper client node could generate. When handling open and transfer transactions, the network was saturated at the 1,000 req/s send rate. So, we select 1,000 req/s as the representative send rate in the following experiments.

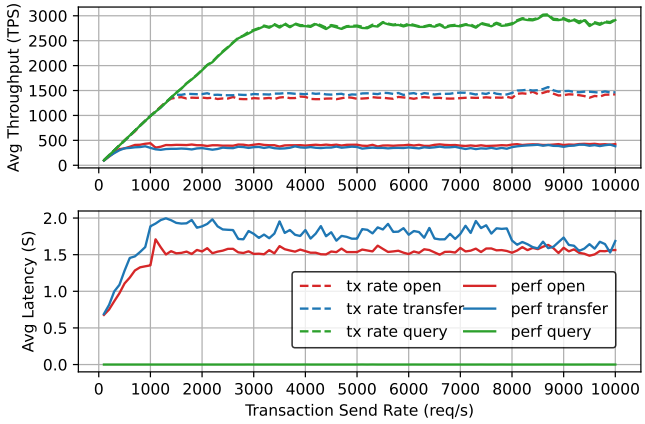

Fig. 2: Besu performance under heavy load with the baseline configurations. Solid lines indicate the average throughput and latency, while dashed lines represent the actual number of requests per second sent to the Besu network.

Figure 3 shows the detailed performance of Besu with the baseline configuration under varying send rates from 100 to 1,000. We can see that the average throughput of open transactions keeps increasing to over 400 TPS with an average latency of less than 1.5 seconds when the send rate reaches 1,000. The performance of open transactions consistently outperforms that of transfer transactions, especially under a heavier workload of more than 500 req/s, where the average throughput is around 10% higher with latency around 10% lower. This subtle difference could be caused by the different complexities of transaction operations. The result also indicates that the query operation is very efficient in Besu, reflected by its close-to-zero latency and linearly increasing throughput.

For the performance trend, the open and transfer throughput increases in a near-linear manner under a low send rate (e.g., from 100 req/s to 500 req/s), where a very small variation can

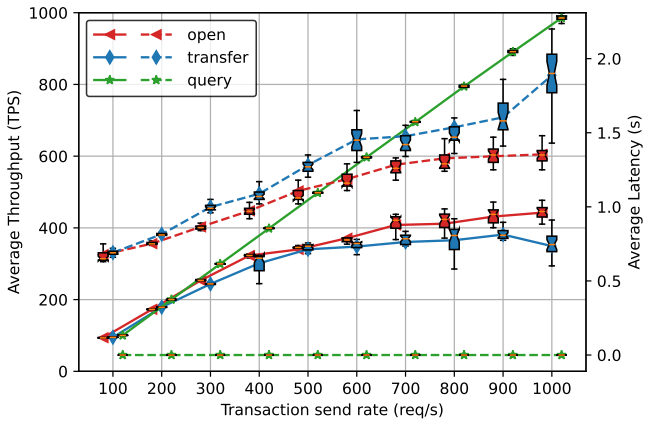

Fig. 3: Besu performance under the baseline configurations. Solid lines indicate throughput; dashed lines represent latency.

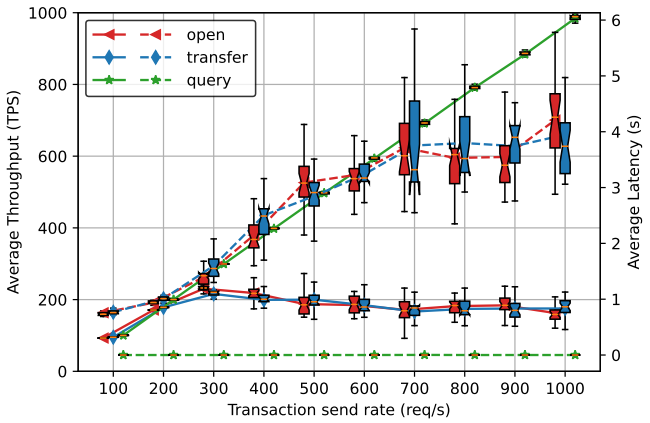

Fig. 4: Besu performance under the baseline configuration without load balancing. Solid lines indicate throughput; dashed lines represent latency.

be seen from the box plot, which implies a stable performance. However, the average throughput increases very slowly or keeps unchanged after the Besu network is saturated at a high send rate (e.g., from 600 req/s to 1,000 req/s), where the result variation becomes larger than light workloads.

B. Load Balancing

To explore the effect of load balancing on the Besu network performance, we tested a Besu network with the same configurations as the baseline, except for the load balancing (LB) setup (Section III-D). Instead of connecting to the load balancer, we directly connected Caliper to a Besu node, which is responsible for both receiving RPC requests and processing consensus tasks. From Figure 4, we can see that the transaction latency of open and transfer at a high request rate is unstable, reflected by large variations in the box plots. By comparing Figure 4 with Figure 3, it is obvious that Besu performance with LB is much better than that without load balancing (NLB). The throughput of open and transfer transactions with LB (i.e., 400TPS) is around twice as high as NLB (i.e., 200TPS). Accordingly, the latency of open/transfer transactions with LB is around half of NLB's. But, the query performance is not impacted by the setting of a load balancer. This is because query operation only reads from one Besu node's ledger and 1,000 req/s is still far away from its maximum RPC capability. In contrast, write (open/transfer) operations involve block creation, consensus message process and transit, transaction validation, etc., which require much more computation power and memory than query. So, a single validator as the RPC node could be easily saturated by an overwhelming number of transaction requests.

As we can see in Table III, in the NLB mode, besu-1 is obviously saturated in terms of CPU and memory utilization at 1,000 req/s send rate, while other nodes have relatively low utilization. The unbalanced load indicates the RPC node resource as a bottleneck of decentralized applications built on top of a Besu network. To make full use of all resources in the network, we add a load balancer to redirect the WebSocket RPC requests from Caliper to all Besu nodes in a round-robin manner. As seen in Table III, the load is balanced among nodes in the LB mode, and the performance of Besu increases as expected by comparing Figure 3 with Figure 4.

C. Scalability

We explore the scalability of the Besu network from two perspectives, scale-out (or horizontal scalability) and scale-up (or vertical scalability). Scale-out is evaluated by varying the network size in a fixed node flavor, i.e., 2C-7.5GB, while scale-up is evaluated by changing the node flavor in a fixed network size (i.e., 8 nodes).

Figure 5 and Figure 6 show the scalability of different consensus algorithms at 1,000 req/s open transaction rate. We observe that QBFT performs slightly better than the other two consensus algorithms in both horizontal and vertical scalability.

Figure 7 shows the detailed horizontal scalability of QBFT. The average performance is getting worse as the network size increases in general. But, it is interesting to see that the open throughput has a small improvement when the network size increases from 4 to 8 before dropping back to less than 400 TPS. Accordingly, the open latency has a subtle decrease at 8 nodes configuration. This is attributed to the increased network size, which distributes the transaction load to more nodes and reduces the queue time on each receiving node. However, as the network size increases to a certain extent that the communication complexity dominates the processing delay and becomes the new bottleneck, the performance gradually drops. We can also observe that open transaction has a consistently better performance than transfer.

Figure 8 shows the detailed vertical scalability of QBFT. We observe that Besu achieves better performance as the network nodes are equipped with more CPU and/or memory resources. The average throughput increases rapidly from 1C7.5G to 4C15G, and then increases slowly as more resources are added to nodes. This is because after reaching a big enough node

TABLE III: Besu node resource utilization statistics and comparison between LB and NLB modes at 1,000 req/s send rate.

	Statistics	besu-1	besu-2	besu-3	besu-4	besu-5	besu-6	besu-7	besu-8
NLB	CPU%(max)	**162.05**	5.57	4.49	3.52	5.43	4.25	3.50	4.00
	CPU%(avg)	**58.24**	4.16	3.24	2.78	4.23	3.39	2.76	3.21
	RAM(max GB)	**2.39**	0.88	0.86	0.93	0.84	0.86	0.86	0.84
	RAM(avg GB)	**2.39**	0.88	0.86	0.93	0.84	0.86	0.86	0.84
LB	CPU%(max)	4.91	5.76	6.23	6.24	6.52	7.37	8.21	5.81
	CPU%(avg)	3.62	3.92	4.29	4.31	4.63	5.39	5.33	4.04
	RAM(max GB)	0.94	0.91	0.98	0.92	1.00	0.96	1.02	0.90
	RAM(avg GB)	0.93	0.90	0.96	0.91	0.99	0.95	1.01	0.90

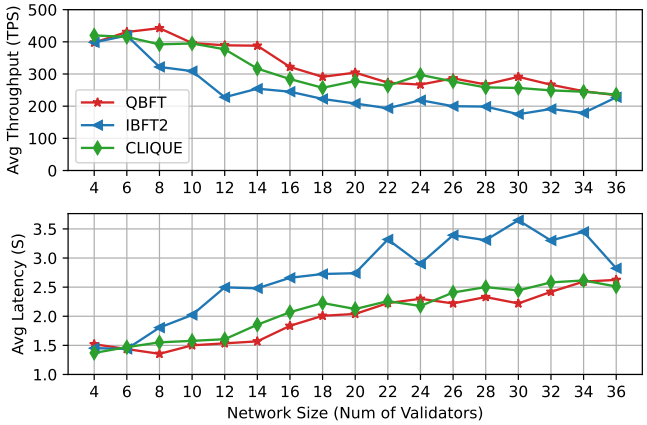

Fig. 5: Horizontal scalability against different PoA consensus algorithms at 1,000 req/s open transaction send rate.

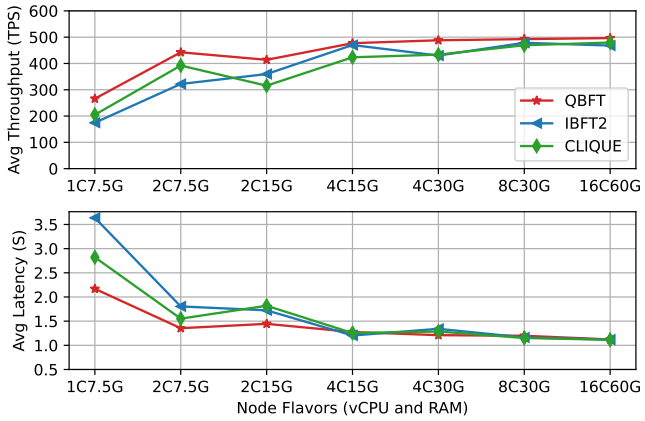

Fig. 6: Vertical scalability against different PoA consensus algorithms at 1,000 req/s open transaction send rate.

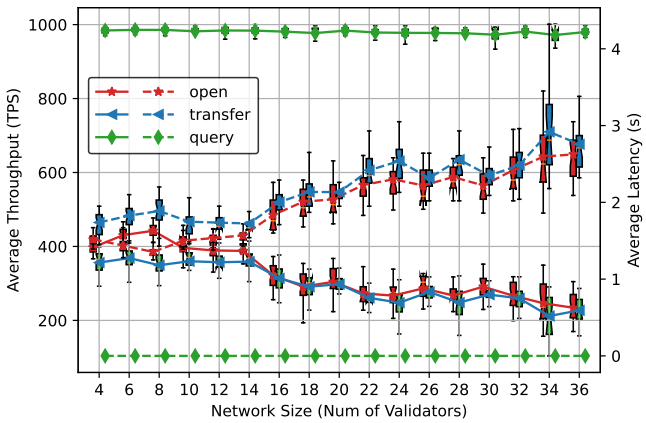

Fig. 7: Horizontal scalability of QBFT at 1,000 req/s send rate. Solid lines indicate throughput; dashed lines represent latency.

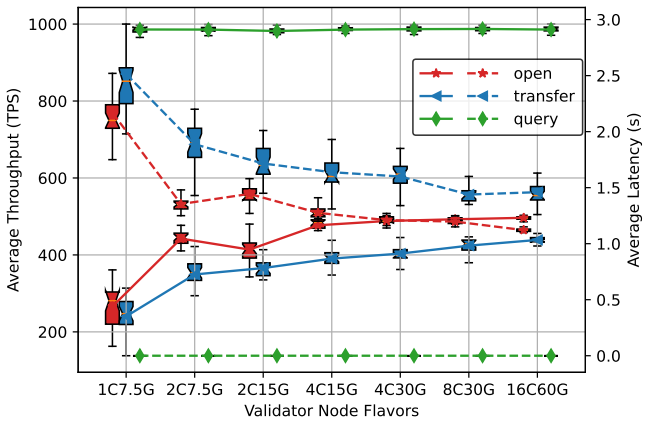

Fig. 8: Vertical scalability of QBFT at 1,000 req/s send rate. Solid lines indicate throughput; dashed lines represent latency.

configuration, the hardware limitation is no longer the main performance bottleneck. Hence, keeping adding resources will have a very limited help on the performance improvement of the Besu network.

D. Block Time

In Besu networks with proof of authority consensus algorithms, block time ("blockperiodseconds" in configuration) determines the frequency of block creation. The timer starts when the protocol receives a new chain head, and when it expires, the protocol proposes a new block. Figure 9 shows how the block time influences the performance of Besu with QBFT consensus. We observe a significant performance drop when the block time increases from 1 to 5 seconds. This is expected since QBFT spends more time waiting for the next block proposal when the configured block time is longer.

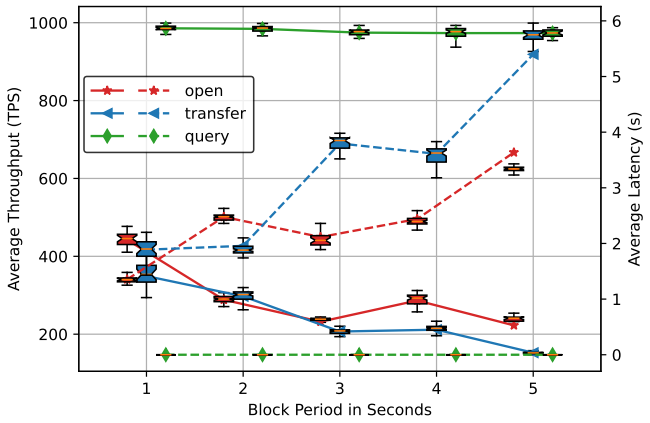

Fig. 9: Performance against varying block times at 1,000 req/s send rate.

E. Consensus

Figure 10 shows the transaction performance of Besu against three proof of authority algorithms, QBFT, IBFT2, and Clique, under varying send rates. We observe that these three consensus algorithms have very close throughput under low workload, e.g., 400 req/s. Then, QBFT has subtly better performance under high send rates, and Clique has a slightly lower throughput than the other two. However, Clique has a relatively lower transfer latency at the same time. Also, the open transaction has a better performance in terms of both throughput and latency in all three consensus algorithms.

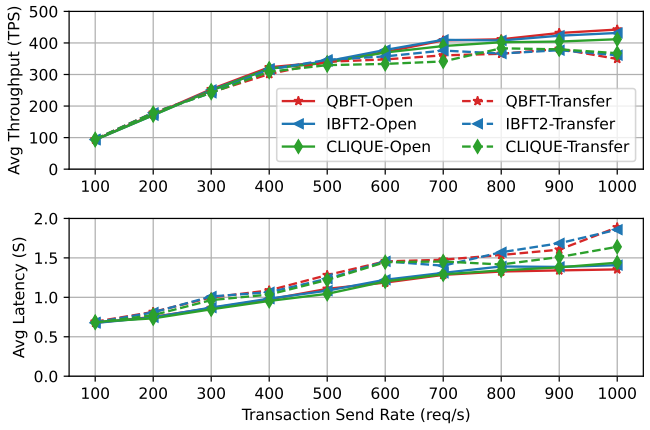

Fig. 10: Performance against different proof of authority consensus algorithms under varying transaction send rates.

V. Analysis and Discussions

In previous sections, we presented the performance benchmark results of the deployed private Hyperledger Besu networks under various configurations. From these results, we have identified some critical parameters which impact the blockchain performance. In this section, we aim to understand and present how these parameters impact Besu performance through log analysis, and discuss the limitations.

Load Balancing: First, a load balancer is significant to improve the system performance for decentralized applications built on a private Besu blockchain. In a public blockchain network, workloads are assumed to be balanced among all the connected nodes because of its fully decentralized nature. But, in a private blockchain network in enterprise applications, a load balancer needs to be considered in system design.

Scalability: The scalability of a vote-based blockchain system has been studied in [20], [21]. Similar to the conclusion in previous works, we found that Besu has limited vertical scalability in terms of node size, while it does not horizontally scale well in terms of an increasing number of validators. When a validator is more powerful, it takes less time to execute transactions, create blocks, and process consensus tasks. By analyzing the logs, we found that a Besu network validator with more computational resources generally has more transactions packed into a single block, namely a bigger block size, at the same block creation frequency and transaction send rate. However, this improvement becomes less and less significant as the node size grows to a high level (e.g., 16C60G), in which the hardware resource is no longer the bottleneck, and it can hardly reduce the computation time any further. For the network size, increasing the number of validators will increase the block propagation and consensus time in a proof of authority blockchain. Since the message transmission is very efficient in a private network, Besu QBFT performance does not have an obvious drop until the network size reaches 16. The tested scalability result of QBFT is lower than the official scalability of IBFT 2.0, which claims that IBFT 2.0 handles up to 30 validators with no loss of performance.

To further understand how the consensus time contributes to latency as network size increases, we extracted the consensus time of all blocks from logs. The consensus time of each non-empty block is calculated by the following equation:

$$T_{\text{consensus}} = TS_{\text{import}} - TS_{\text{create}} \tag{1}$$

where TS_{import} is the timestamp of the last node importing the block, and TS_{create} is the timestamp of block creation. Note that only open and transfer transactions contribute to the consensus time as query operation is not involved in any consensus process. Figure 11 shows the box plots of QBFT consensus time and the quorum size when the number of validators increases from 4 to 36. We observe that the consensus time increases slowly as network size increases before reaching 30. Then, it has a significant growth from 30 to 34. The mean consensus time is even over one second at 34 nodes configuration. By comparing this figure with Figure 7, we can see that as the number of nodes increases, the consensus time occupies a higher and higher proportion of the entire transaction latency.

Figure 12 shows the consensus time of Besu QBFT under various node flavor configurations. We can observe that the consensus time decreases when node flavor changes from a

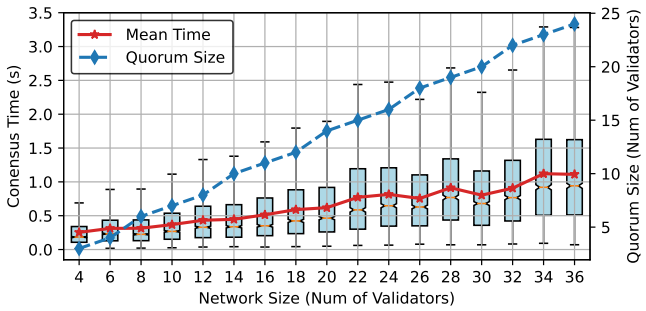

Fig. 11: Besu QBFT consensus time and quorum size against varying network size. Quorum size is the minimum number of validators voting on a block in a stage so that the consensus can proceed to the next stage.

small size to the baseline configuration, then it has a subtle improvement even increased to an extra large instance, i.e., 16 vCPU and 60GB RAM. This indicates that node computation resources have a limited impact on consensus time in a Besu network with a fixed number of validators, e.g., 8. A Besu node with adequate resources, for example, 2vCPU and 7.5GB RAM, can efficiently create blocks, process consensus messages, and update states to the database in the QBFT consensus.

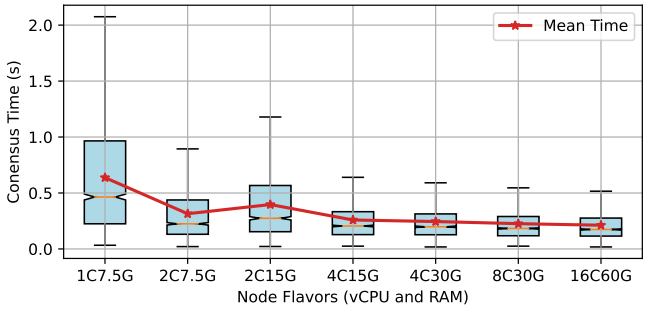

Fig. 12: QBFT consensus time against varying node flavors.

Block Time: Another critical parameter, block time, determines the block frequency, which directly impacts the latency and throughput of Besu. This parameter should be carefully configured according to the application's needs in practice. On the one hand, if a block period is very short (e.g., 1s), Besu will generate blocks very frequently. This results in high throughput and low latency for consensus algorithms with finality properties such as QBFT and IBFT 2.0. However, a short block period will cause more forks in a Clique network, which in turn degrades the performance. In addition, a lot of empty blocks will be created in the case of low transaction loads, resulting in a low storage utilization. On the other hand, if a block period is very long, transactions need to wait for a long time to be packed into the next block, which significantly reduces Besu performance. In a small private network, the block creation and processing times are the prominent components of transaction latency. Unlike Geth [20], the performance of Besu is not inversely linear against the increasing block period seconds. This indicates that Besu is not as efficient as Geth in creating and processing blocks, so that the validator is not capable of packing all the received workload (1,000 transactions) into a single block within a short period (e.g., one second). Instead, the rest are pending and waiting for the next block.

Block Size has a direct impact on the throughput of the Besu network by determining the number of transactions in a block. Figure 13 shows the box plots of block size against varying transaction send rates under the baseline configurations, where Caliper sent 100,000 open and transfer transactions in total; out of them, 99,432 were executed successfully and wrapped into 383 blocks. The mean block size increases linearly from 100 to 400 req/s send rate, then grows slowly as the transaction send rate increases. This trend keeps aligned with the baseline throughput in Figure 3. We can also see larger size variations and many outliers starting from 500 req/s transaction send rate. This is because proposers can not process a large number of transactions and wrap them all into the current block in one second, leaving a small portion to be processed in the next block.

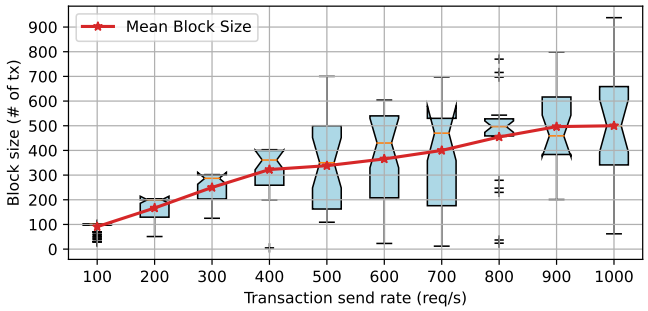

Fig. 13: Block size against varying transaction send rates.

Consensus: Different types of PoA consensus algorithms (i.e., Clique, IBFT 2.0, and QBFT) in Besu have a very similar performance profile in our experiments. Theoretically, Clique should reach consensus and add blocks faster than IBFT 2.0 and QBFT under the same configuration, since the latter two require three extra communication stages before adding blocks to the chain. However, for Clique, the probability of a fork and out-of-order sealing increases as the number of validators increases, especially at a small block period. From our experiments, three PoA consensus algorithms have similar performance, which indicates that consensus time cost is negligible compared to other operations such as transaction execution and blockchain state updates in a small private blockchain network with 8 nodes.

Transaction Types: At last, workload type is one of the most important factors influencing the Besu network performance. This can be seen in all our experiments that open transaction consistently has a better performance than transfer. The root cause is that transfer transactions have two write

operations, while open transactions only have one. Therefore, under the same node configuration, Besu can execute more open transactions than transfer and pack them into a block, which increases the block size and improves the performance.

Limitations: All the findings and conclusions are constrained by specific experimental setups on the OpenStack cloud, which limits the generalization. For example, since all VMs are deployed on the same server cluster with high-speed communication among nodes, the network delay is negligible. In contrast, a real production deployment may experience higher latency and lower transaction throughput under the same node configurations. In addition, the load balancer may incur additional development concerns about session persistence and node affinity in practice.

VI. Related Work

Many studies have conducted performance analyses on various private blockchain networks [22], mainly focusing on Hyperledger Fabric (HLF) and private Ethereum. Most of these studies use Hyperledger Caliper [18] to benchmark and evaluate the performance.

Performance analysis of Hyperledger Fabric was discussed in [23]–[29]. Baliga *et al.* [23] studied the throughput and latency of HLF v1.0 using Caliper by tuning transaction and chaincode configuration parameters, such as the number of chaincodes, channels, and peers. Thakkar *et al.* [24] explored the performance bottlenecks of the HLF v1.0 under different block sizes, endorsement policies, number of channels, resource allocation, and state database choices (GoLevelDB v.s. CouchDB). Based on the identified bottlenecks, the authors suggested three optimization solutions, i.e., parallel VSCC validation, cache for membership service provider (MSP), and bulk read/write for CouchDB, which were implemented in HFL v1.1. Androulaki *et al.* [25] explored the impact of block size, peer CPU, and SSD v.s. RAM disk on blockchain latency, throughput, and scalability under different numbers of peers in HLF v1.1. Nguyen *et al.* [26] conducted an experimental study to explore the impact of large network delays on the performance of HLF v1.2.1 over an area network between France and Germany. In [27], Kuzlu *et al.* analyzed throughput, latency, and scalability of Hyperledger Fabric v1.4, the first long-term support release, under varying transaction send rates and participants. Meanwhile, Geyer *et al.* [28] used Caliper to benchmark HLF by tuning network parameters, e.g., latency and packet loss, and to explore the influence of transaction rate, chaincode, network properties, local network impairment, and block size. Wang [29] researched the performance of HLF under abnormal behaviors by designing multiple malicious behavior patterns and testing the transaction throughput and latency in such contexts. The results show that delay attacks, along with keeping some replicas out of working, dramatically decrease the system performance.

Performance analysis of private Ethereum platform was discussed in [20], [21], [30], [31]. In [30], Rouhani and Deters analyzed two most popular Ethereum clients, Geth and Parity, in a private network. The results indicate that, compared to PoW-based Geth, PoA-based Parity is 89.82% faster in transaction processing, on average, under different workloads. In [20], Schaffer *et al.* introduced a concept for measuring the performance and scalability of private Ethereum smart contract platforms to examine the impact of various parameters on Ethereum performance. In [31], Bez *et al.* analyzed the scalability of Ethereum under an extensible test environment with synthetic benchmarks. In a recent work [21], Samuel *et al.* proposed a performance test tool by integrating a load-balancer middleware, to comparatively evaluate three mainstream Ethereum clients with their proof of authority consensus algorithms, namely Geth 1.9.18 with Clique, OpenEthereum(Parity) V3.0.1, and Hyperledger Besu 1.5.0 with Clique and IBFT 2.0, for private blockchain. Unfortunately, they only evaluated the throughput and scalability in several configurations. In addition, due to the limitations of their queue and nonce management design, the tested throughput in load balancing was even worse than the non-balanced results.

Some other performance evaluation of private blockchain was discussed in [32], [33]. In [32], Arati *et al.* comparatively studied the throughput and latency characteristics of Quorum, a permissioned blockchain platform built from the Ethereum codebase, under Raft and IBFT consensus, respectively. Shi *et al.* [33] empirically studied the performance on consistency, stability, and scalability of Sawtooth, another well-known permissioned blockchain platform from Hyperledger.

Performance evaluation of private blockchain in the context of peer-to-peer transaction applications was discussed in [9], [10], [34]. Lohachab *et al.* [34] proposed a secure peer-to-peer energy trading framework, called HFPET, based on Hyperledger Fabric. They evaluated two different versions (i.e., v1.4 and v1.4.1) of Fabric through extensive experimentation using Caliper. In [10], Pradhan *et al.* proposed a flexible permission ascription (FPA) based Hyperledger Besu IBFT 2.0 for peer-to-peer energy trading, and conducted comparative performance evaluation between Besu and other blockchain platforms, including Hyperledger Fabric Raft and Kafka, and Ethereum Ethash. In another recent work [9], Abdella *et al.* compared the performance of a proposed Hyperledger Besu IBFT-based system, called UBETA, with other three blockchain systems (i.e., Ethereum Clique, Ethereum PoW, and Hyperledger Fabric Raft) using specific performance metrics. However, the authors used their own test metrics and tool rather than the popularly adopted Caliper benchmarking method, which might impact the confidence in the results.

Different from all the above studies, our work is the first one to research the performance of QBFT consensus, and systematically study the performance of Hyperledger Besu for enterprise applications.

VII. Conclusion

In this work, we introduced a load balance-based blockchain performance evaluation architecture and leveraged it to benchmark the Hyperledger Besu private blockchain. We designed and conducted extensive experiments to evaluate the impact of

selected parameters that may affect the network performance, such as transaction send rate, network size, node flavor, load balancing, consensus, and block time. Our results indicate that blockchain parameters such as block time and block size have the most significant impact. In addition, Besu performance is limited by block creation and transaction execution, which are determined by various factors, such as node configuration, transaction complexity, and load balancing. We also find that QBFT has slightly better scalability than IBFT 2.0 and Clique in Besu. It can scale up to 14 validators without obvious performance loss. All these findings, along with root cause analysis, have paved the way for further analysis and performance improvement of Hyperledger Besu. For future work, it will be interesting to investigate the private transaction performance of the Enterprise Ethereum blockchain and compare the performance of different clients implementing the same consensus. In addition, we would like to build an analytical model to characterize the performance of Besu.

ACKNOWLEDGEMENT

This work has been partially supported by Huawei-ECE Research Initiative at the University of Alberta. Digital Research Alliance of Canada (alliancecan.ca) and Cybera (cybera.ca) partially supported this research work through their cloud computing resources.

REFERENCES

[1] R. Casado-Vara, J. Prieto, F. De la Prieta, and J. M. Corchado, "How blockchain improves the supply chain: Case study alimentary supply chain," *Procedia computer science*, vol. 134, pp. 393–398, 2018.

[2] H. Wu, J. Cao, Y. Yang, C. L. Tung, S. Jiang, B. Tang, Y. Liu, X. Wang, and Y. Deng, "Data management in supply chain using blockchain: Challenges and a case study," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–8, IEEE, 2019.

[3] A. Shahid, A. Almogren, N. Javaid, F. A. Al-Zahrani, M. Zuair, and M. Alam, "Blockchain-based agri-food supply chain: A complete solution," *IEEE Access*, vol. 8, pp. 69230–69243, 2020.

[4] G. Leeming, J. Cunningham, and J. Ainsworth, "A ledger of me: personalizing healthcare using blockchain technology," *Frontiers in medicine*, vol. 6, p. 171, 2019.

[5] S. Tanwar, K. Parekh, and R. Evans, "Blockchain-based electronic healthcare record system for healthcare 4.0 applications," *Journal of Information Security and Applications*, vol. 50, p. 102407, 2020.

[6] S. Wang, A. F. Taha, J. Wang, K. Kvaternik, and A. Hahn, "Energy crowdsourcing and peer-to-peer energy trading in blockchain-enabled smart grids," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 8, pp. 1612–1623, 2019.

[7] S. Seven, G. Yao, A. Soran, A. Onen, and S. Muyeen, "Peer-to-peer energy trading in virtual power plant based on blockchain smart contracts," *IEEE Access*, vol. 8, pp. 175713–175726, 2020.

[8] A. Esmat, M. de Vos, Y. Ghiassi-Farrokhfal, P. Palensky, and D. Epema, "A novel decentralized platform for peer-to-peer energy trading market with blockchain technology," *Applied Energy*, vol. 282, p. 116123, 2021.

[9] J. Abdella, Z. Tari, A. Anwar, A. Mahmood, and F. Han, "An architecture and performance evaluation of blockchain-based peer-to-peer energy trading," *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3364–3378, 2021.

[10] N. R. Pradhan, A. P. Singh, N. Kumar, M. Hassan, and D. Roy, "A flexible permission ascription (fpa) based blockchain framework for peer-to-peer energy trading with performance evaluation," *IEEE Transactions on Industrial Informatics*, 2021.

[11] Péter Szilágyi, "EIP-225: Clique proof-of-authority consensus protocol." https://eips.ethereum.org/EIPS/eip-225, March 2017. no. 225.

[12] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, pp. 173–186, 1999.

[13] AMIS Team, "Istanbul Byzantine Fault Tolerance." https://github.com/ethereum/EIPs/issues/650, 2017. no. 650.

[14] R. Saltini and D. Hyland-Wood, "Correctness analysis of ibft," *arXiv preprint arXiv:1901.07160*, 2019.

[15] R. Saltini and D. Hyland-Wood, "Ibft 2.0: A safe and live variation of the ibft blockchain consensus protocol for eventually synchronous networks," *arXiv preprint arXiv:1909.10194*, 2019.

[16] "Istanbul BFT's design cannot successfully tolerate fail-stop failures." https://github.com/ConsenSys/quorum/issues/305, 2018.

[17] H. Moniz, "The istanbul bft consensus algorithm," *arXiv preprint arXiv:2002.03613*, 2020.

[18] Performance and S. W. Group, "Hyperledger blockchain performance metrics white paper," tech. rep., HyperLedger Found., 2018.

[19] Caliper Benchmarks Maintenance Team, "Simple Contract." https://github.com/hyperledger/caliper-benchmarks/blob/main/src/ethereum/simple/simple.sol, 2019.

[20] M. Schäffer, M. d. Angelo, and G. Salzer, "Performance and scalability of private ethereum blockchains," in *International Conference on Business Process Management*, pp. 103–118, Springer, 2019.

[21] C. N. Samuel, S. Glock, F. Verdier, and P. Guitton-Ouhamou, "Choice of ethereum clients for private blockchain: Assessment from proof of authority perspective," in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1–5, IEEE, 2021.

[22] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance evaluation of blockchain systems: A systematic survey," *IEEE Access*, vol. 8, pp. 126927–126950, 2020.

[23] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat, and S. Chatterjee, "Performance characterization of hyperledger fabric," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 65–74, IEEE, 2018.

[24] P. Thakkar, S. Nathan, and B. Viswanathan, "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, IEEE, 2018.

[25] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15, 2018.

[26] T. S. L. Nguyen, G. Jourjon, M. Potop-Butucaru, and K. L. Thai, "Impact of network delays on hyperledger fabric," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 222–227, IEEE, 2019.

[27] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: throughput, latency and scalability," in *2019 IEEE international conference on blockchain (Blockchain)*, pp. 536–540, IEEE, 2019.

[28] F. Geyer, H. Kinkelin, H. Leppelsack, S. Liebald, D. Scholz, G. Carle, and D. Schupke, "Performance perspective on private distributed ledger technologies for industrial networks," in *2019 International Conference on Networked Systems (NetSys)*, pp. 1–8, IEEE, 2019.

[29] S. Wang, "Performance evaluation of hyperledger fabric with malicious behavior," in *International Conference on Blockchain*, pp. 211–219, Springer, 2019.

[30] S. Rouhani and R. Deters, "Performance analysis of ethereum transactions in private blockchain," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 70–74, IEEE, 2017.

[31] M. Bez, G. Fornari, and T. Vardanega, "The scalability challenge of ethereum: An initial quantitative analysis," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 167–176, IEEE, 2019.

[32] A. Baliga, I. Subhod, P. Kamat, and S. Chatterjee, "Performance evaluation of the quorum blockchain platform," *arXiv preprint arXiv:1809.03421*, 2018.

[33] Z. Shi, H. Zhou, Y. Hu, S. Jayachander, C. de Laat, and Z. Zhao, "Operating permissioned blockchain in clouds: A performance study of hyperledger sawtooth," in *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, pp. 50–57, IEEE, 2019.

[34] A. Lohachab, S. Garg, B. H. Kang, and M. B. Amin, "Performance evaluation of hyperledger fabric-enabled framework for pervasive peer-to-peer energy trading in smart cyber–physical systems," *Future Generation Computer Systems*, vol. 118, pp. 392–416, 2021.