BlockMeter: An Application-Agnostic Performance Measurement Framework for Private Blockchain Platforms

Ifteher Alom, Md Sadek Ferdous, Member, IEEE, and Mohammad Jabeed Morshed Chowdhury, Member, IEEE

Abstract—Blockchain Technology is an emerging technology with the potential to disrupt a number of application domains. Though blockchain platforms like Bitcoin and Ethereum have seen immense success and acceptability, their nature of being public and anonymous makes them unsuitable for many enterprise-level use-cases. To address this issue, Linux Foundation has started an open-source umbrella initiative, known as the Hyperledger Platforms. Under this initiative, a number of private blockchain platforms have been developed. However, the scalability and performance of these private blockchains must be examined to understand their suitability for different use-cases. Recent researches and projects on performance benchmarking for private blockchain systems are specific to use cases and generally tied to a blockchain platform. In this article, we present BlockMeter, an application-agnostic performance benchmarking framework for private blockchain platforms. BlockMeter can be utilised to measure the key performance metrics of any application deployed on top of an external private blockchain application in real-time. In this article, we present the architecture of the framework and discuss its different implementation aspects. Then, to showcase the applicability of the framework, we use BlockMeter to evaluate the two most widely used Hyperledger platforms, Hyperledger Fabric and Hyperledger Sawtooth, against a number of use-cases.

Index Terms—Blockchain, performance, performance benchmarking, performance metrics, scalability.

I. INTRODUCTION

The concept of Blockchain Technology is undeniably an ingenious invention of Satoshi Nakamoto [1]. Equipped with a smartly engineered combination of distributed computing and cryptographic mechanisms [2], [3], Blockchain technology has the potential to be one of the frontier technologies in the very near future. To investigate its potentiality is being actively investigated, within academia, industries and governments around the world, how blockchain technology can be integrated with the existing use cases or even how it can be leveraged to disrupt the traditional application domains, such as banking, finance, e-government, healthcare industries, IoT, agriculture and so on [4], [5]. Many industries in these application domains are investigating in shifting their infrastructures towards blockchain.

To cater to this need, in addition to Bitcoin, a number of new blockchain platforms, such as Ethereum [6], Cardano, Algorand [7], Polkadot [8] and others have emerged. These blockchain platforms are equipped with novel features such as smart contracts and offer additional advantages in terms of scalability and performance in comparison to Bitcoin. However, these platforms are public in nature, meaning anyone can verify every single block, even observe every single activity and submit transactions anonymously or pseudonymously which makes them unsuitable for enterprise-level applications where privacy and identity are key requirements. In addition, there are issues with respect to the scalability and performance of these blockchain platforms for any large-scale adoption in real-life settings [9], [10], [11], [12], [13].

To address these issues, Linux Foundation [14], a non-profit technology consortium, has started an open-source umbrella initiative, known as the Hyperledger Platforms. The goal of this initiative is to achieve an industry-wide collaboration for developing enterprise-grade blockchain platforms. Under this initiative, a number of private Blockchain platforms have been developed which can be used for different applications. However, to utilise and select a particular blockchain platform for a specific use case, one must be able to compare the available platform against a set of criteria and select the best suitable one.

One of the key factors driving the selection of a specific blockchain platform for an application is performance. Recently, there have been a number of research and projects on performance benchmarking of private blockchain systems [9], [10], [15], [16]. However, these projects are specific to use cases and are generally tied to a blockchain platform. This means that the method presented in those works cannot be used to evaluate the suitability of any application for a specific blockchain platform. Furthermore, many of the experiments presented in those works have been conducted in simulated environments. Hyperledger Project itself has a tool called Caliper [17], unfortunately, it cannot measure the performance of blockchain applications deployed externally. To mitigate these limitations, in this article, we present BlockMeter, an application-agnostic, real-time performance benchmarking framework for private blockchain...
The major contributions of this article are

- We compare different performance metrics for private blockchain platforms.
- We present the architecture of BlockMeter, an application-agnostic blockchain performance measurement platform.
- We discuss different implementation aspects of BlockMeter and describe how it can be integrated with different use cases.
- To showcase the applicability of BlockMeter, we utilise BlockMeter to evaluate the two most widely used Hyperledger platforms, Hyperledger Fabric and Hyperledger Sawtooth, against a number of use cases and blockchain network configurations.
- Finally, we present a comparative analysis of BlockMeter and the existing relevant research works and the Caliper tool against a set of features, highlighting the benefits of BlockMeter over the existing tools.

Structure: In Section II, we briefly discuss blockchain and its different aspects and explain different terminologies used in our research and the blockchain platforms on which we have conducted our experiments. Section III discusses briefly some of the relevant research works within the scope of this article. In Section IV, we present the high-level architecture of BlockMeter and discuss its implementation methodologies and process flow. In Section V, we present the conducted experiments and their corresponding results and in Section VI we compare different types of applications and their performance with respect to two blockchain platforms. We discuss different aspects related to BlockMeter in Section VII. Finally, in Section VIII, we conclude our article with a hint of future goals and expectations.

II. BACKGROUND

In this section, some of the relevant key concepts are discussed. In the following, we briefly describe blockchain (Section II-A), the Hyperledger projects (Section II-B), Hyperledger Fabric (Section II-C), Hyperledger Sawtooth (Section II-D) and Hyperledger Caliper (Section II-E).

A. Blockchain

In the year 2008, a digital currency named Bitcoin was introduced by someone with the presumed pseudonym Satoshi Nakamoto [2]. Bitcoin promised the ability to carry out financial transactions without relying on a central authority (e.g., government-backed central bank), utilising different cryptographic mechanisms. All transactions are kept on a public ledger or the blockchain, ensuring transparency [21]. Bitcoin’s main technological breakthrough is due to its underlying technology called blockchain. A blockchain is an immutable append-only, ordered data structure, called ledger, consisting of many blocks of data linked together by cryptographic protocols [3]. Each block contains some transactions where each transaction is a record of an action undertaken by a user to transact a certain amount of currency/data to another user/users. Each block refers to its previous block using a cryptographic hash, which refers to its previous block and so on, hence forming a chain which is colloquially known as blockchain.

Despite the vast number of blockchain platforms and implementations available at this moment, we can broadly classify them into two types: permissionless (public) and permissioned (private) [2]. A public blockchain allows any user to participate in the blockchain network and block generations process, submit transactions and validate transactions and blocks. On the other hand, a private blockchain acts as a closed ecosystem, where only authorised users are able to join the network, see the recorded history, or issue transactions. Indeed, such blockchains are governed by specific members of a consortium or organisation and only the approved members and computer entities have the possibility of running nodes on the network, validating transaction blocks, issuing transactions, executing smart contracts or reading the transaction history [19]. Bitcoin [20] and Ethereum (Main-net) [6] are examples of public blockchains, whereas Hyperledger platforms [21] and Quorum [22] are examples of private blockchain systems.

As the popularity of Bitcoin, Ethereum and a few other derivative technologies grew, interest in applying the underlying technology of the blockchain and distributed ledger to more innovative enterprise use cases also grew. However, many enterprise use cases require performance characteristics that the currently available public blockchain platforms are unable to deliver [11], [12], [13]. In addition, in many use cases, the identity of the participants is a hard requirement, such as in the case of financial transactions where Know-Your-Customer (KYC) and Anti-Money Laundering (AML) regulations must be followed [23]. Furthermore, the privacy of transactions and data is an important factor. Therefore, in the case of enterprise/business applications and use cases, mostly permissioned blockchains are utilised to restrict its access within the members of an organisation or consortium.

B. Hyperledger

Hyperledger is an open-source collaborative project undertaken to advance cross-industry blockchain technologies [24]. It is a global collaboration, hosted by The Linux Foundation, which includes leaders from finance, banking, the Internet of Things, supply chains, manufacturing and Technology. The Hyperledger project aims to accelerate industry-wide collaborations for developing high-performance and reliable blockchain and distributed ledger-based tools, standards and guidelines that could be used across various industry sectors to enhance the efficiency, and performance of various business processes [25]. The various projects under the Hyperledger umbrella include among others, the following:

- **Hyperledger Fabric** [23]: an enterprise-grade, permissioned blockchain platform for building various blockchain-based products, solutions and applications for business use cases.
- **Hyperledger Sawtooth** [26]: an enterprise-level, permissioned, modular blockchain platform for building
blockchain applications and networks using an innovative Proof of Elapsed Time [27] consensus algorithm.

- **Hyperledger Caliper [17]**: a blockchain benchmark tool that is used to evaluate the performance of a specific blockchain implementation.

Next, we present a few more details about these projects which are relevant to this article.

### C. Hyperledger Fabric

Hyperledger Fabric [28] is a permissioned blockchain framework, offering a highly modular and configurable architecture. Being permissioned means that the participants within the same blockchain network are known to each other, rather than anonymous and the trust requirement is dependent upon the application developed on top of it.

One crucial feature of Fabric is the support of general-purpose programming languages for deploying smart contracts (a computer program consisting of the business logic, deployed and executed on top of a computing platform underpinned by a blockchain), known as *chaincode* in Fabric terminology. Currently, Fabric supports several languages such as Golang, JavaScript, Java and others [29]. Hyperledger Fabric has different types of nodes. The committing peers maintain the ledger by holding a copy of it, while the endorsing peers are responsible for saving a copy of the ledger as well as receiving requests from the clients and validating them against the chaincode and endorsement policies (rules that govern who is allowed to do what within the blockchain platform). Finally, the orderers are responsible for batching and maintaining the order of transactions over the entire network. Since multiple organisations may participate in the same Fabric network, the Membership Service Provider (MSP) provides each peer with cryptographic identities and each organisation have a Certification Authority (CA) [15].

Hyperledger Fabric incorporates a new concept called channels, which is an instance of the blockchain with its own chaincode that can be accessed only by a predefined subset of participants. A transaction in Hyperledger Fabric is initiated by the client by submitting a transaction proposal that has been created using the respective SDK (Software Development Kit). The proposal is broadcasted to the endorsing peers according to the policies to verify the validity and integrity of the transaction. The endorsing peers simulate the transaction and respond with a signed proposal response. The proposal is then sent to the ordering service where the transactions are combined into blocks which are transmitted to all peers. All peers commit this block to the blockchain [23]. Fig. 2 presents a high-level overview of this process.

### D. Hyperledger Sawtooth

Hyperledger Sawtooth [26] is another blockchain platform under the Hyperledger ecosystem. It has been designed to be highly scalable and efficient in terms of performance. Sawtooth supports different consensus protocols [30] like Practical Byzantine Fault Tolerance (PBFT) and Proof of Elapsed Time (PoET) which depend on Intel Software Guard Extension (Intel SGX). Intel SGX is a Trusted Execution Environment (TEE)

built into some Intel central processing units (CPUs). They allow user-level and operating system code to define protected private regions of memory, called enclaves. SGX allows the processing of instructions within a secure enclave inside the processor [31], [32], [33], [34].

The architecture of Sawtooth consists of four main components: a validator, a transaction processor, a consensus engine and a REST API. The validator node is the heart of the system that is responsible for handling transactions, and generating and storing blocks. The transaction processor node holds the business logic for validating transactions. The consensus engine uses an algorithm for ordering the transactions into blocks. The REST API node receives transactions from the clients and communicates with the validator nodes for getting the responses.

The transaction flow in Sawtooth (illustrated in Fig. 1), begins with the client submitting transactions to the validator node using the REST API. The SDK provides necessary functionalities to compile the transaction(s) into a block, also known as batches, then sign them and finally, to send them to a validator. The validator broadcasts the batch to the transaction processors to validate their integrity and commits the transactions. All transactions within a batch are committed to the state together or not at all [35].

### E. Hyperledger Caliper

Hyperledger Caliper [17] is a performance benchmark tool under the Hyperledger umbrella. It provides a generic method for the performance evaluation of several blockchain technologies.
Hyperledger Caliper has a layered architecture that allows the separation of benchmark parameters from the ledger implementation. The generic benchmark layer monitors the resource consumption, evaluates the recorded data and generates a report at the end of the test. Blockchain interfaces allow different blockchain implementations to communicate with the benchmark layer. The adaptation layer consists of necessary modules for different ledger implementations [15]. The adaptation layer is used to integrate any existing blockchain system into the Caliper framework using network configuration files.

The benchmarking criteria in Caliper so far include success rate, transaction throughput, transaction latency and resource consumption. However, the main limitation of using Hyperledger Caliper for benchmarking or for analysing the performance of a blockchain application is that we cannot execute the test operations on an externally deployed blockchain platform. The Caliper requires the platform specifications to be fed into its architecture and it deploys a blockchain platform accordingly. However, in real life, blockchain platforms may be deployed in a complex system environment (for example using docker containers, using virtual machines connected remotely, over a local organisation network and so on) which cannot be replicated in Caliper. In addition, Caliper only tests the performance of the underlying blockchain platform, completely detaching the overlay application and thus, it is difficult to benchmark a fully integrated application using Caliper. Because of all these factors, it is difficult to create an application-agnostic benchmarking framework for real-life deployments using Caliper.

F. Performance Metrics

There are a number of parameters which can be utilised to compare and analyse the performance of blockchain systems. From our research and references from existing works as discussed in Section III, we have identified a number of such parameters that are mostly used in private blockchain platforms and these parameters are throughput, latency and resource consumption. Next, we present a brief discussion of these parameters.

Throughput is defined as the number of transactions that are successfully completed and committed to the ledger in unit time. Throughput is determined by the capacity of the network. In other words, the network infrastructure must be sufficient to handle the traffic. Theoretically, throughput will increase as the input load is increased, up to the maximum network capacity [36]. However, network capacity, apart from physical setup, also depends on the internal logic execution and access control mechanism.

Latency can be defined as the time elapsed between a transaction submission and completion. Latency is dependent on propagation delay, transit and queuing of the system. The protocol flow of the system plays a crucial role in either increasing or decreasing the latency. High latency is a major issue with most of the established blockchain platforms like Bitcoin [37].

Resource consumption is referred to as the extent to which resources such as the CPU and main memory of the host system are used by different nodes of the blockchain network during its execution. A blockchain system requires a network of nodes which have different roles and perform various operations in order to maintain the integrity of the network and ledger. Despite the various benefits that a blockchain platform provides, one of the major subjects of discussion is its significant requirements for resources. Public blockchain systems such as Bitcoin are maintained by power and resource-hungry mining rigs [38] which is a reason businesses are looking for private blockchain systems that are more efficient in this regard [39].

A Workload can be defined as the load that is executed on a blockchain platform to evaluate its performance and its ability to handle a specific use case. Elementary workloads aim to focus on a single aspect of the network. These include DoNothing (minimal chaincode functionality: read/write), CPUHeavy (heavy computation chaincode) and DataHeavy (chaincode involving large size data and parameters). These generic workloads are simulated implementations of realistic use cases. For example, Fabcar (a demo application supplied with the Fabric codebase), which performs read, write and insert operations on a predefined set of records using key-value pairs can be taken in the category of a DoNothing chaincode.

III. RELATED WORK

The performance analysis of private blockchain platforms is a relatively new dimension of research, however, there are already a few works in this domain. In this section, we briefly review and analyse the existing works.

Leppelsack et al. have described a generic implementation of a framework that can measure the performance of Hyperledger Fabric under different workloads using different chaincode [15]. To simulate the workload, Hyperledger Caliper has been used. Caliper deploys the network, installs the chaincode and submits transactions at a predefined rate and reports the latency and resource consumption of different nodes. Four different experiments have been conducted using a Fabric network consisting of a single orderer and two organisations, each having two peers. The impact of varying transaction rates, chaincode, block size and network loss have been studied graphically.

The authors in [9], have demonstrated the development and analysis of the first private blockchain evaluation framework –
Blockbench. The authors have used the framework to study the performance of three different private blockchains—Ethereum, Parity and Hyperledger Fabric under different workloads. According to their experimental results, the current state and performance of private blockchain systems are not mature enough to replace the existing database systems. Blockbench framework allows the integration of various blockchain platforms via simple Application Programming Interfaces (API) which is also a feature of our project. Different kinds of chaincode have been used to simulate various use cases. The measurement metrics include throughput, latency, resource consumption of the nodes, and scalability. The evaluation mainly focuses on the comparison of different blockchain implementations based on the metrics. The impact of ledger-specific configurations (e.g. block-size) has not been considered.

The authors in [10], have presented an evaluation of Hyperledger Fabric and Ethereum. For both of platforms, they have implemented an application with different smart contracts. The execution times of the smart contracts are recorded and compared. However, the main focus lies in the performance of these blockchain platforms by varying the number of transactions that are executed. Metrics on the ledger layer including execution time, transaction latency and transaction throughput have been analysed. These metrics are evaluated for both platforms with varying numbers of transactions.

The experiments show the differences in execution time of a varying number of transactions, with different platforms and different functions. The execution time increases as the number of transactions in the data set increases. The execution time of Hyperledger Fabric is consistently lower than Ethereum and the gap between the execution time of Fabric and Ethereum also grows larger as the number of transactions increases. As the number of transactions in the data set grows, the latency of transactions in Ethereum worsens considerably more compared to Hyperledger Fabric. It has been also found that Fabric has higher throughput than Ethereum in all of the experimental data sets. However, for similar computational resources, Ethereum can handle more concurrent transactions.

The authors in [16] have conducted a comprehensive empirical two-phase study to understand and optimise the performance of Hyperledger Fabric by varying configuration parameters such as block size, endorsement policy, channels, resource allocation, and state database choices. In the first phase, the impacts of these Fabric configuration parameters on the transaction throughput and latency have been studied. It has been found that endorsement policy verification, sequential policy validation of transactions in a block, and state validation and commit (with CouchDB) have been the three major bottlenecks. In the second phase, they focused on optimising Fabric v1.0 based on the observations gained in the first phase. Several optimisations have been introduced and studied, such as aggressive caching and parallelising for endorsement policy verification in the cryptography component and bulk read/write optimisation for CouchDB during state validation.

All the research works discussed so far closely match with the scope of the current research in sense their main focus is to analyse the performance of private blockchain system. In addition to these works, many blockchain based research works also carried out different performance related experiments and reported them in their research works. In the following we have reviewed a few research works from different application domains.

In [40], the authors presented a blockchain and IoT-based hybrid infrastructure for transportation insurance. The insurance premium-related services are encapsulated in “on-chain” chaincodes to perform over data on the vehicle’s trip and driver’s behaviour, which are collected through “off-chain” analytic services using the sensing data collected from vehicles’ onboard sensors. In the article, they also presented a performance evaluation of their developed prototype and suggested that properly defined channel configuration and efficient state database can be used to improve the performance of large-scale applications. The authors in [41] presented a network model for blockchain-based IoT systems and theoretically analysed the throughput performance. In [42], investigated how blockchain technology could be extended to the application of vehicle networking and distributed secure storage of Big Data. Based on the simulation of the proposed theoretical model in Matlab, the authors discussed data transmission performance over a cellular network.

In [43], [44], the authors discussed a blockchain-based solution for medical records management between patients and healthcare institutions. Hyperledger Caliper has been used to capture performance metrics like throughput, latency and resource usage. The authors in [45] discussed a Personal Health Record (PHR) model based on openEHR standard specifications [46] and emphasised the importance of performance analysis for the adoption of blockchain technology. The proposed model was tested locally using manually configured computers and Apache JMeter [47]. In [48], the authors presented a privacy-preserving e-health system using a blockchain-based model. Their proposal leveraged the private blockchain for storing personal health data and a consortium blockchain to keep records of the data sharing. The system was evaluated on JUICE [49] (an open platform for designing Ethereum smart-contract-based applications) for performance evaluation.

The literature review clearly underlines the importance of performance evaluation for several application domains where blockchain technology can be utilised to provide a decentralised and secure solution to existing methods. After reviewing these research works, we have identified two trends:

- Most of these research works focused on throughput and latency as key metrics for performance analysis. BlockMeter has been designed to handle such metrics as well.
- There were no standard ways to measure the performance of blockchain-based applications. In order to introduce a coherent approach, an application-agnostic tool like BlockMeter would be crucial.

IV. ARCHITECTURE, IMPLEMENTATION AND PROCESS FLOW

The core theme of BlockMeter is that it advocates the concept of an application agnostic performance measurement framework for private blockchain systems. Towards this aim, in this section, we present the architecture of BlockMeter (Section IV-A) which illustrates how different components of the architecture interact for a particular testing scenario. Based on the architecture, we...
have developed a Proof of Concept (PoC) in order to evaluate its applicability. The implementation details for the PoC are described in Section IV-B. Section IV-C presents the process flow for utilising BlockMeter for testing and evaluations.

A. Architecture

The architecture, as described in Fig. 3, consists of several modules. Each module performs a specific task by receiving some data, applying a set of instructions to it and then forwarding it to the next module. Next, we elucidate the architecture and its different components.

- **Blockchain Application:** This module comprises the use-case or client application under test. It is responsible for setting up a blockchain-based application and its constituent elements independently.

- **API Gateway:** In order to serve requests from the client application each supported blockchain platform has been assigned a particular path where the client requests are received and relayed to the transaction processor (discussed later). Finally, upon the completion of the request, the response is sent back to the client.

- **Performance Data:** Upon the completion of each round of load testing we receive the data recorded by the performance monitor. The data consists of request start times and end times, throughput count and system statistics which includes memory usage, percentage of CPU usage, network traffic and disc read-write) from the docker containers that host the blockchain application. These data can be parsed and studied for further analysis based on the requirements and criteria of evaluation.

- **Performance Monitor:** This can be called the crux of the system as it is responsible for collecting the performance data as the transactions are being cascaded from the framework to the blockchain and vice versa. Several methods are embodied in this module to collect data like transaction start time, end time, and the status of the docker containers. The data thus collected are continually uploaded to specific files in the directory, which can be retrieved for further analysis.

- **Transactions Handler:** It is one of the central components that receives incoming requests and initiates the process. Its main role is to translate incoming HTTP requests into blockchain platform-specific data-frames. Every blockchain platform has its own prerequisites and conditional input fields that are used at various levels of the transaction processing, this module encapsulates the incoming request into a blockchain object. At the end of the process, when the transaction is completed, the object is parsed to get the necessary information and the client is notified with a response.

- **Transaction Processors:** This component translates client requests into objects that can be perceived by a particular blockchain platform within its ecosystem. The respective Software Development Kit (SDK) for each blockchain platform provides the necessary functions and libraries to achieve this task. In our framework, we have exploited the SDK to develop such a middleware for Hyperledger Fabric and Hyperledger Sawtooth platforms. More blockchain platforms can be incorporated here to make the support range wider and more comprehensive.

- **Blockchain Platform:** The backend of the platform that contains the business logic in the chaincode/smart contract and the network topology of the hosting environment is represented by the blockchain platform. The available resources and the complexity of validation policies play a crucial role in the performance of the application.

B. Implementation

In order to implement the PoC based on the architecture, we would need to integrate different private blockchain platforms with the framework. From the available private blockchain...
platforms, we have selected Hyperledger Fabric and Sawtooth from the Hyperledger Projects [24] for the PoC as these two are popular private blockchain platforms. The implementation instance of the developed PoC based on the proposed architecture of BlockMeter is illustrated in Fig. 3.

At the Blockchain Application level (Blockchain Application part of Section IV and in Fig. 3), we have configured two blockchain networks as discussed before. The Hyperledger Fabric network consists of a number of different types of nodes as discussed earlier. Similarly, for the Sawtooth network, required validating nodes and transaction processor nodes have been set up. For both cases, the blockchain application have been deployed using Docker (used for virtualisation). The network topology and structure have been varied for different experiments (discussed in Section V).

The API endpoints are exposed from a NodeJs-Express-based server that is responsible for handling incoming traffic and routing to the respective blockchain middleware. We have used the NodeJS SDK of Fabric and Sawtooth in our implementation to create a middleware (the Transaction Processor in Section IV), which accepts client requests and compiles them into platform-specific objects that can communicate with the backend blockchain network using the respective SDK. The Performance Monitor extends several NodeJs libraries to fetch system information to monitor docker containers.

BlockMeter is designed with the intention of testing private blockchain platforms. The entire system (blockchain application and the BlockMeter framework) is intended to be deployed within the local network of an organisation. Moreover, during the testing/experiments, we do not use or access real payload and signing keys. Hence, the implementation of BlockMeter does not discuss access and privacy control measures. However, user authentication and SSL [50] protocols can be easily integrated with the API Gateway of BlockMeter framework.

C. Process Flow

Now, we illustrate a process flow utilising the framework in Fig. 4. This flow showcases the interactions between different components of the architecture when used in a testing scenario.

- **User Creation:** A number of user accounts have been created for each blockchain platform. The user credentials consist of the user names and their respective public and private keys that are used to sign the transaction requests before submitting them to the blockchain platform. These accounts have been used to simulate scenarios where a number of users simultaneously use the application built on top of that particular blockchain platform.

- **Payload Creation:** In order to perform an operation in the blockchain (i.e. invoking a particular function of the smart contract in the blockchain) we need to provide the necessary parameters. In this step, such parameters are created programmatically with random payloads.

- **Traffic/Load Generation:** The payload and parameters generated earlier are imported into Apache JMeter [47] that injects them into HTTP requests and hits the framework’s Gateway API. JMeter is an open source software quality evaluation tool maintained by the Apache Software Foundation [51]. It is primarily used for load testing, unit testing and basic performance monitoring of web-based applications. Using JMeter, the number of requests/loads generated per second is varied over the span of each experiment.

- **Transaction Request:** As illustrated in Fig. 4, the JMeter acts as the source to generate transaction requests. These requests are essentially HTTP requests consisting of transaction particulars for the respective blockchain platform. We can feed in different configuration parameters, e.g. the number of simulated users, the frequency of user requests and so on, from a pre-loaded CSV. JMeter submits these user requests to the API Gateway. Upon the receipt of HTTP requests at the API gateway of the framework, an instance of each request is created and performance metrics are initialised and attached to it. Each request is validated for valid parameters and the data is normalised into the required structure.

- **Request Pre-processing:** Once the transaction parameters are retrieved from the requests, the API gateway forwards these to the Transactions Handler where an instance of the transaction is created and the structured data object is sent to the corresponding transaction processor of the blockchain platform, where the request is pre-processed as per the platform requirements. This includes the creation...
of a signed transaction instance that consists of the payload and execution instructions.

- **Performance Monitoring:** Once a transaction instance is ready to be submitted to the blockchain, the performance monitor is notified with a signal to start recording performance metrics and the usage status of docker containers/instances that are hosting the blockchain network. Finally, the Performance Data consisting of transaction latency, transaction throughput and resource consumption are recorded in JSON format into the file system. The records are then normalised, reformatted and converted into a tabular format using NodeJS [52] which are then visually analysed using python libraries.

- **Blockchain Application:** The transaction request is received by the blockchain application backend. The business logic of the application is executed and the process is completed with a success response.

V. EXPERIMENTS

To show the applicability of the developed performance measurement framework, we have designed and carried out a number of experiments. In this section, we present the details of various aspects of the experiments conducted using the framework.

A. Experimental Setup

The principal motivation of the experiments is to showcase how the developed framework can be utilised to gauge the performance, in terms of throughput, latency, and resource consumption, of different private blockchain platforms, Hyperledger Fabric and Sawtooth in this instance.

Towards this aim, we have conducted several experiments and load testing by simulating some of the typical use cases under varying testing conditions (like low and high request traffic and a number of operating nodes of the blockchain platform). To conduct these experiments, our system has been deployed in Amazon Web Services (AWS) EC2 [53] instances powered by 4 vCPUs and 16 Gigabytes of RAM. The different blockchain network configurations have been initiated with various orderer nodes using docker in AWS EC2 instances as well.

Every use case involves a streamlined flow of operations (refer to Protocol Flow in Section IV-C) that is initiated/requested by the Apache JMeter in the experiment, followed by a set of pre-processing at the framework and finally, concluded by appending the transaction to a blockchain. Next, we present a brief overview of the courses of action involved in this regard.

In order to demonstrate a comparison of the performance metrics between the chosen blockchain platforms, we have selected three different types of applications that are exemplary of typical user applications.

- **Simple Application:** A simple key-value store that stores certain values against unique IDs within the blockchain. This application performs operations such as inserting, updating and delete on the data. It is representative of a typical read/write application.

- **Data Heavy Application:** This application handles a large amount of data in every transaction. It performs read and write operations on records that are 10 Kilobytes in size. It represents a use case of a data-intensive application.

- **CPU Heavy Application:** In this application we have modified the chaincode to introduce a CPU-intensive task consisting of repeated arithmetic operations on random numbers. Every transaction is constrained to perform the mathematical operation before it comes to completion.

These three applications represent three broad categories of applications that may use blockchain in their backend data validation and processing. These findings will give us an insight into the performance of Hyperledger Fabric and Hyperledger Sawtooth when use cases require the handling of some intense tasks.

In a nutshell, the procedure involved in a single round of an experiment began by importing randomly generated payload at Apache JMeter which was then translated into a transaction and submitted to the blockchain platform while tracing the performance in terms of latency, throughput and resource consumption at the same time. For different scenarios and different simultaneous cycles, the rate of incoming traffic or the load varied steadily. For Hyperledger Fabric, experiments began at 10 requests per second that gradually increased to 1000. For Hyperledger Sawtooth, we began at 10 transactions per second (tps), but had to wind up at 600 tps (details discussed in Section VII). At the end of the experiments, the data recorded by the performance monitor is represented using graphs to facilitate a comprehensive performance analysis of the underlying platform.

B. Findings

In this section, we explore the overall performance of Hyperledger Fabric and Hyperledger Sawtooth in terms of throughput, latency and resource consumption under varying numbers of operating nodes (orderers and peers in Fabric and validators and processors in Sawtooth).

**Hyperledger Fabric:** The performance results of the experiments involving Hyperledger Fabric have been plotted in Fig. 5. An important observation from Fig. 5(a) is that the transaction throughput of Fabric has increased steadily with increasing traffic. However, the trend levelled off and began to decrease when the input traffic load crossed 600 per second. The increase in the number of operating nodes (ordering service and peers) did result in an increase of the throughput slightly for every single transaction load group. The primary reason for such behaviour is that a typical blockchain transaction has to filter through a number of security policies and requirements before it is appended to the main blockchain. This intricate secure mechanism of the blockchain system incurs a delay and backlog which results in a comparatively lesser throughput under higher traffic.

The latency in the transaction response increases significantly as the number of transactions arriving per second increases (Fig. 5(b)). The situation did not improve even when multiple ordering service and peer nodes are added to the network. The time in seconds clearly demonstrates that we need more optimisation in the architecture of the Fabric platform. The
complex architecture and protocol flow of the Fabric system can be accounted for the delayed response. Moreover, more operational nodes mean more synchronising overhead which may overrun the added computation. However, there are several tweaks that we can make: like setting the block size and block creation timeout that may improve the response time depending upon the situation and type of application [16].

Regarding the CPU usage, it can be seen in Fig. 5(c) that the processor reaches its capacity sooner with fewer ordering nodes. Memory usage increases as the number of transactions arriving per second increases. In Fabric, peer nodes consume more memory as they perform a significant amount of tasks for validation as well as store a copy of the blockchain. As a consequence, its memory consumption is higher and keeps growing as more transactions are appended. Multiple ordering nodes further increase memory usage as seen in Fig. 5(d).

Hyperledger Sawtooth: It can be observed from Fig. 5(e) that transaction throughput has increased gradually with increasing traffic. Also, multiple validating nodes lead to a slight improvement in the throughput performance. Hyperledger Sawtooth uses a hardware-dependent consensus algorithm known as Proof of Elapsed Time (PoET) that relies upon the Software Guard Extension (SGX) capability of the processor [31], [32], [33], [34]. The working principle of the PoET algorithm involves
To summarise, BlockMeter has the following advantages:

- **CPU Heavy** application is designed as a middleware between the CPU and the transaction load. With the addition of multiple validating nodes and transaction processors, the delay is improved slightly. However, the results reveal that Sawtooth is way behind Hyperledger Fabric when it comes to response latency. As highlighted earlier, the consensus algorithm of Sawtooth and the underlying mechanism that is reliant on SGX facility and leader election is the key reason for such delayed transaction processing. More importantly, this excessive backlog has accounted for almost a system freeze when inbound traffic reached 800 requests per second and higher.

- **Data Heavy** application shows a higher rate of memory consumption than Fabric. Fig. 5(h) illustrates that memory usage increases almost steadily as the transaction arrival rate is increased. Multiple validator nodes further increase resource availability and usage.

VI. COMPARISON OF DIFFERENT APPLICATIONS

With respect to the throughput of Hyperledger Fabric (Fig. 5(i)), Simple application provides the maximum throughput as expected. The CPU Heavy application gives the minimum throughput and the Data Heavy application shows a slightly better output. As observed earlier in Section V-B, we can observe the throughput falling when input traffic exceeds 600 transactions per second. As expected, in case of latency (Fig. 5(j)), CPU Heavy application has the highest response delay, followed by Data Heavy application and simple application. The CPU Heavy application being overwhelmed with intensive computation and the Data Heavy application loaded with large size parameters, such behaviours are understandable. What is striking though is that these numbers might deter enterprise application development where a quicker response may be a key requirement.

On the other hand, Hyperledger Sawtooth has shown a much-deteriorating result in the case of both throughput and latency. Although throughput has increased gradually with increasing traffic (Fig. 5(k)), the latency has increased drastically (Fig. 5(l)). It can almost certainly be said that applications at the production environment require a much better output for these areas of performance. It is observed that Simple, Data Heavy and CPU Heavy with increasing payload size and logical complexity respectively, accounted for slightly poorer performance. As discussed in Section V-B, the consensus algorithm of the Sawtooth platform is a major reason for this poor performance. It goes without saying Sawtooth blockchain framework has a lot of scope for improvement in comparison to Fabric.

VII. DISCUSSION

As blockchain technology and its areas of applications in the mainstream are being heavily contemplated, it is crucial to have a methodology to efficiently measure its practicality in terms of performance under various conditions. In the course of our research, we have realised that performance measurement and analysis in the domain of blockchain, have been primarily limited to public blockchain platforms like Bitcoin [20] and Ethereum [6]. Very few research works have considered private blockchain platforms and in them, they have compared private blockchain platforms with popular public blockchains, which is not very fruitful as the two systems function quite differently and are intended for dissimilar audiences.

This article presents BlockMeter, an application-agnostic performance measurement framework for private blockchain platforms. The application-agnostic feature stems from the idea that it is not coupled with any particular application. Indeed, any blockchain application can utilise the framework to measure and compare the performance of different private blockchain platforms for that particular application by submitting/simulating user interactions via any load testing software such as Apache JMeter. This has been possible as our framework exposes different APIs which act as the entry point for outside applications to interact with the system. The PoC that we have developed currently measures the performance of Hyperledger Fabric and Hyperledger Sawtooth.

From the different related work and analysis, it has been found that latency and throughput are the two key decisive elements of performance in the field of blockchain applications. As such, we have focused more on these metrics to determine the performance of the applications in our experiments. With the advent of silicon chips, memory and computing capability have skyrocketed in recent times. In addition, power and energy consumption are directly linked to resource usability and efficiency of a computer system. Hence, we have also incorporated a resource usage recording mechanism within BlockMeter, allowing us to consider resource usage as a performance determinant.

**Advantages:** To summarise, BlockMeter has the following advantages:

- BlockMeter is capable of measuring performance metrics of an independently deployed blockchain application. The recorded metrics are key indicators of the system’s performance and can act as determining factors of its applicability for a particular business application.
- The performance framework can be easily integrated with applications developed using Hyperledger Fabric and Hyperledger Sawtooth platforms. The application-specific configuration is minimal as the main execution system (i.e. the blockchain and its configuration) is independent of the framework.
- BlockMeter is designed as a middleware between the transaction invocation and completion to allow application-agnostic performance testing. Hence it enables us to test different logical aspects and implementations of chaincode by invoking chaincode functions with appropriate payload. Moreover, it helps any blockchain-based application to load test their applications against a number of users and blockchain network settings.
- Our tests have been conducted by hosting BlockMeter and deploying the blockchain applications on Amazon AWS EC2 [53] machines. This has helped us evaluate the
performance in a more realistic environment in contrast to a controlled local test setting.

- BlockMeter has been designed in a modular fashion that allows more blockchain platforms to be easily integrated into it.

The findings of our experiments show similar trends as compared to some related work [16], [55] and confirm the validity of the results of BlockMeter. However, there are significant differences in the design, process flow and test environment of BlockMeter with other related works, hence, it is not practical to fully match the results with them.

Comparative analysis: Next, we analyse how BlockMeter fares in comparison to other relevant works discussed in Section III. The comparative analysis is summarised in Table I against a number of criteria. Table II highlights the key points that give BlockMeter an upper hand over Caliper [54]. In the tables, the symbol ‘•’ has been used to denote certain criteria are satisfied by the respective work whereas the symbol ‘○’ indicates that a certain criterion is not satisfied.

As it can be seen from Table I, BlockMeter uses an API-based approach for ensuring that it is not tied to a particular application, thus facilitating its application-agnostic feature. In contrast, the top-level applications are tightly coupled in most of the works. This means significant changes need to be made whenever a new application is integrated into their systems. Also, BlockMeter provides a modular framework, giving it the flexibility to add additional blockchain platforms easily. This is also absent in most of the works.

The evaluation metrics provide the key insights from a performance evaluation framework. Most of the related research has focused only on latency and throughput for their analysis while others also considered resource consumption. However, BlockMeter is the only one which considered all three evaluations metrics. BlockMeter also considered three different types of applications as per many works. BlockMeter is also equipped with a performance monitoring mechanism that makes it self-sufficient for recording performance data, whereas most of the related work had to rely on Caliper [54] or some other tools.

Also, most of the research has been conducted in local servers or controlled environments which eliminates much overhead that arises in real-life environments. For example, some works deployed their implementations within local servers. The BlockMeter framework and the blockchain platforms have been deployed in AWS EC2 [53] virtual machines via web service settings which allowed us to perform the tests in a more realistic setting. Moreover, some of the research has used Hyperledger Caliper [56] or general load testing tools for the evaluation which limits the scope of the experiment as the target blockchain is deployed internally by the testing tool itself, rather than a realistic and independent externally deployed platform.

Next, we present a comparative analysis between Caliper [54] and BlockMeter. The result of the analysis is summarised in Table II where the symbols denote the semantics discussed previously. From the table, it can be observed that BlockMeter provides a number of advantages and benefits in comparison to Caliper. To begin with, Caliper is designed with a much
more tightly-coupled layer of components that limit its flexibility to be modified and configured to specific needs for a particular application. On the contrary, BlockMeter provides a very modular architecture that enables anyone to easily configure it for any application. Furthermore, BlockMeter provides an API gateway for any application to interact with the underlying blockchain framework, thus facilitating an application-agnostic performance evaluation. Also, unlike Caliper, the components of BlockMeter are designed to support easier integration with other blockchain platforms beyond the Hyperledger project. However, BlockMeter and Caliper have certain common aspects as well. Both are capable of monitoring similar performance metrics and generate tabular summaries based on the recorded experiment data.

**Limitations:** Our developed BlockMeter framework is capable of measuring crucial performance metrics of private blockchain platforms, nevertheless, it has some limitations.

- The current implementation is limited to only two private blockchain platforms, Hyperledger Fabric and Hyperledger Sawtooth. There are other private blockchain platforms emerging as well. However, we have designed the framework in such a way that other blockchains with similar architectural setups can be easily integrated and we have plans for full-fledged support for other Hyperledger projects [25].

- Our existing implementation does not provide a central dashboard with the summary of the executed tests. An automated visual interface would be very helpful for any end user to get primary insights of the test results.

- Our implementation is generally focused on the comparison and analysis of private blockchain platforms with are more favourable for enterprise applications. However, public blockchains are also being used by a huge number of users. We would also like to explore the possibility of adding a sub-module for crypto-currency-based blockchains to study and analyse their performance.

**VIII. Conclusion**

Blockchain is a promising technology that aims to solve many of the security and integrity issues of our traditional data processing systems. In recent times, a great deal of interest has been observed within the academia and industry which is a result of frequent research and the popularity of blockchain platforms like Bitcoin and Ethereum. As a result, a blockchain is being considered a viable option in multiple application domains including government, finance, telecommunications and others. One of the major obstacles in the acceptability of blockchain is how it might perform in different application domains. Also, how different blockchain platforms would fair against each other in a number of criteria. This is particularly true for private blockchain platforms which require enterprise-level performances [11], [12], [13]. In order to address this issue, in this article, we have presented BlockMeter, an application-agnostic performance measurement framework, that enables to compare the performance of different private blockchain platforms, against some metrics, for any given application. To show the applicability of our framework, we have evaluated the performance of two popular private blockchain platforms, Hyperledger Fabric and Hyperledger Sawtooth under different applications with different configuration parameters. From our experimental findings, we have found that Fabric performs slightly better than Sawtooth under a very high request rate.

As of now, we have integrated support for two blockchain platforms from the Hyperledger projects. We intend to incorporate other private blockchain platforms from the Hyperledger project and beyond and experiment with them to understand how they would perform against each other. We believe that this research will be a foundation for many performance evaluation tools. Such an evaluation tool could be an important component for many enterprises that are willing to integrate their applications with a blockchain platform.

**References**

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Decentralized Bus. Rev., 2008.
[2] M. S. Ferdous, M. J. M. Chowdhury, and M. A. Hoque, “A survey of consensus algorithms in public blockchain systems for crypto-currencies,” J. Netw. Comput. Appl., vol. 182, 2021, Art. no. 103035.
[3] M. J. M. Chowdhury et al., “A comparative analysis of distributed ledger technology platforms,” IEEE Access, vol. 7, no. 1, pp. 167 930–167 943, 2019.
[4] M. J. M. Chowdhury, M. S. Ferdous, K. Biswas, N. Chowdhury, and V. Muthukumarasamy, “A survey on blockchain-based platforms for IoT use-cases,” Knowl. Eng. Rev., vol. 35, 2020, Art. no. e19.
[5] M. S. Ferdous, K. Biswas, M. J. M. Chowdhury, N. Chowdhury, and V. Muthukumarasamy, “Integrated platforms for blockchain enablement,” in Advances in Computers. Amsterdam, Netherlands: Elsevier, 2019, pp. 41–72.
[6] Ethereum, Accessed: Jul. 10, 2020. [Online]. Available: https://www.ethereum.org
[7] Algorand, “Algorand developer docs,” Accessed: Dec. 12, 2021. [Online]. Available: https://developer.algorand.org/docs/
[8] Polkadot, “Getting started,” Accessed: Dec. 27, 2021. [Online]. Available: https://wiki.polkadot.network/docs/getting-started.
[9] T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, “BLOCKBENCH: A framework for analyzing private blockchains,” in Proc. ACM Int. Conf. Manage. Data., 2017, pp. 1085–1100.
[10] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, “Performance analysis of private blockchain platforms in varying workloads,” in Proc. 26th Int. Conf. Comput. Commun. Netw., 2017, pp. 1–6.
[11] J. Yu-Huono, D. Ko, S. Choi, S. Park, and K. Smolander, “Where is current research on blockchain technology?—a systematic review,” PLoS One, vol. 11, no. 10, 2016, Art. no. 0163477.
[12] J. Mendling et al., “Blockchains for business process management-challenges and opportunities,” ACM Trans. Manage. Inf. Syst., vol. 9, no. 1, pp. 1–16, 2018.
[13] H. Fabric, “Introduction,” Accessed: Oct. 10, 2022. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html
[14] L. Foundation, “Case study: Hyperledger,” Accessed: Dec. 27, 2021. [Online]. Available: https://www.linuxfoundation.org/tools/hyperledger/
[15] H. F. Leppelsack, “Experimental performance evaluation of private distributed ledger implementations,” Master’s thesis, Depart. Inform., Tech. Univ. Munich, Munich, Germany, 2018.
[16] P. Thakkar, S. Nathan, and B. Viswanathan, “Performance benchmarking and optimizing hyperledger fabric blockchain platform,” in Proc. IEEE 26th Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst., 2018, pp. 264–276.
[17] H. Foundation, “Hyperledger caliper,” Accessed: Jan. 05, 2022. [Online]. Available: http://github.com/hyperledger/hyperledger-caliper
[18] When and why was Bitcoin Created?, “Plus500,” Jul. 2021, Accessed: Jul. 31, 2021. [Online]. Available: https://www.plus500.com/Instruments/BTCUSD/The-History-of-Bitcoin
[19] Blockonomi, “Permissioned versus permissionless blockchains: Understanding the differences,” Accessed: Jun. 29, 2019. [Online]. Available: https://blockonomi.com/permissioned-vs-permissionless-blockchains
[20] Bitcoin, Accessed: Jul. 10, 2020. [Online]. Available: https://www.bitcoin.org

[21] Hyperledger, Accessed: Jul. 10, 2020. [Online]. Available: https://www.hyperledger.org

[22] Quorum, “Quorum blockchain,” Accessed: Jul. 10, 2020. [Online]. Available: https://www.quorum.com

[23] H. Fabric, “Introduction,” Accessed: Jun. 29, 2019. [Online] Available: https://hyperledger-fabric.readthedocs.io/en/release-1.2/whatis.html

[24] R. Herwanto, H. Sabita, and F. Armawan, “Measuring throughput and latency of blockchain consensus algorithms: A survey,” 2020, arXiv:2001.0709.

[25] Investopedia, “Hyperledger,” Accessed: Jun. 29, 2019. [Online]. Available: https://www.investopedia.com/terms/h/hyperledger.asp

[26] H. Sawtooth, “Introduction,” Accessed: Dec. 26, 2021. [Online] Available: https://sawtooth.hyperledger.org/docs/core/releases/1.0.0rc7/introduction.html

[27] Investopedia, “Proof of elapsed time (cryptocurrency),” Accessed: Feb. 03, 2020. [Online]. Available: https://www.investopedia.com/terms/p/proof-elapsed-time-cryptocurrency.asp

[28] H. Foundation, “Hyperledger fabric,” Accessed: Jan. 05, 2022. [Online]. Available: https://www.hyperledger.org/fabric

[29] H. Fabric, “Hyperledger fabric SDKs,” Accessed: May 05, 2022. [Online]. Available: https://www.hyperledger.org/display/Hyperledger+Fabric+SDKs

[30] H. Sawtooth, “Overview,” Accessed: Dec. 26, 2021. [Online] Available: https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html#dynamic-consensus

[31] Intel, “What is intel SGX?,” Accessed: Oct. 20, 2021. [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html

[32] M. S. Ferdous, M. J. M. Chowdhury, M. A. Hoque, and A. Colman, “Blockchain consensus algorithms: A survey,” 2020, arXiv:2001.0709.

[33] Wikipedia, “Software guard extensions,” Accessed: Jan. 27, 2020–01-27. [Online]. Available: https://en.wikipedia.org/wiki/Software_Guard_Extensions

[34] Wikipedia, “Software guard extensions,” Accessed: Oct. 30, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Software_Guard_Extensions

[35] B. Ampel, M. Patton, and H. Chen, “Performance modeling of hyperledger sawtooth blockchain,” in Proc. IEEE Int. Conf. Intel. Secur. Informat., 2019, pp. 59–61.

[36] R. Herwanto, H. Sabita, and F. Armawan, “Measuring throughput and latency distributed ledger technology: Hyperledger,” J. Inf. Technol. Ampera, vol. 2, no. 1, pp. 17–31, 2021.

[37] R. Yasaweerasinghelage, M. Staples, and I. Weber, “Predicting latency of blockchain-based systems using architectural modelling and simulation,” in Proc. IEEE Int. Conf. Softw. Architecture, 2017, pp. 253–256.

[38] V. Juričić, M. Radošević, and E. Fuzul, “Optimizing the resource consumption of blockchain technology in business systems,” Bus. Syst. Res.: Int. J. Soc. Advancing Innov. Res. Economy, vol. 11, no. 3, pp. 78–92, 2020.

[39] IBM, “Blockchain explained,” Accessed: Jan. 07, 2022. [Online]. Available: https://www.ibm.com/blogs/blockchain/2017/05/the-difference-between-public-and-private-blockchain

[40] Z. Xiao et al., “Blockchain and IoT for insurance: A case study and cyberinfrastructure solution on fine-grained transportation insurance,” IEEE Trans. Computat. Social Syst., vol. 7, no. 6, pp. 1409–1422, Dec. 2020.

[41] Y. Sun, L. Zhang, G. Feng, B. Yang, B. Cao, and M. A. Imran, “Blockchain-enabled wireless Internet of Things: Performance analysis and optimal communication node deployment,” IEEE Internet Things J., vol. 6, no. 3, pp. 5755–5802, Jun. 2019.

[42] T. Jiang, H. Fang, and H. Wang, “Blockchain-based internet of vehicles: Distributed network architecture and performance analysis,” IEEE Internet Things J., vol. 6, no. 3, pp. 4640–4649, Jun. 2018.

[43] G. Al-Sumiaidee, R. Alkhudary, Z. Zilic, and A. Swidan, “Performance analysis of a private blockchain network built on hyperledger fabric for healthcare,” Inf. Process. Manage., vol. 60, no. 2, 2023. Art. no. 103160.

[44] A. Koushik, B. Jain, N. Menon, D. Lohia, S. Chandhari, and V. K, BP, “Performance analysis of blockchain-based medical records management system,” in Proc. 4th Int. Conf. Recent Trends Electron. Inf. Commun. Technol., 2019, pp. 985–989.

[45] A. Roehrs, C. A. da Costa, R. da Rosa Righi, V. F. da Silva, J. R. Goldim, and D. C. Schmidt, “Analyzing the performance of a blockchain-based personal health record implementation,” J. Biomed. Inform., vol. 92, 2019, Art. no. 103140.

[46] openEHR, “What is openEHR?,” Accessed: Jun. 10, 2023. [Online]. Available: https://www.openehr.org/

[47] A. Software, “Apache jmeter,” Accessed: Jul. 13, 2019. [Online]. Available: https://jmeter.apache.org/

[48] A. Zhang and X. Lin, “Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain,” J. Med. Syst., vol. 42, no. 8, 2018, Art. no. 140.

[49] JUICE, Accessed: Jun. 10, 2023. [Online]. Available: https://www.juzhen.io

[50] IBM, “Secure sockets layer (SSL) protocol,” Accessed: Oct. 10, 2022. [Online]. Available: https://www.ibm.com/docs/en/ibm-http-server/0.9.0?topic=communications-secure-sockets-layer-ssl-protocol

[51] T. A. S. Foundation, “Projects lists,” Accessed: Jan. 07, 2022. [Online]. Available: https://www.apache.org/index.html#projects-list

[52] Node.js, Accessed: Jul. 10, 2020. [Online] Available: https://nodejs.org/en/

[53] Amazon, “Amazon EC2 features,” Accessed: Oct. 20, 2021. [Online]. Available: https://aws.amazon.com/ec2/features/

[54] H. Foundation, “Hyperledger caliper,” Accessed: Jan. 05, 2022. [Online]. Available: https://hyperledger.github.io/caliper/v0.4.2/getting-started/

[55] R. Han, G. Shapiro, V. Gramoli, and X. Xu, “On the performance of distributed ledgers for Internet of Things,” Internet Things, vol. 10, 2020, Art. no. 100087.

[56] H. Caliper, “Architecture,” Accessed: Jul. 12, 2019. [Online] Available: https://hyperledger.github.io/caliper/docs/2_Architecture.html

Ifteh Alom received the bachelor’s degree from the Shahjalal University of Science and Technology, Sylhet, Bangladesh. Currently he is working as a software engineer with Samsung R&D Institute, Dhaka, Bangladesh. His research interests include application of blockchain technology, cryptography, and Internet of Things.

Md Sadek Ferdous (Member, IEEE) received the PhD degree in identity management from the University of Glasgow. He is an associate professor with BRAC University, Dhaka, Bangladesh. He is also a research associate with the Centre for Financial Technology of Imperial College Business School, Imperial College London. He has several years of experience of working as a postdoctoral Researcher in different universities in different European and U.K.-funded research projects. He has co-authored more than 60 research papers published in top journals, conferences and workshops.

Mohammad Jabeed Marshed Chowdhury (Member, IEEE) received the PhD degree from the Swinburne University of Technology, Australia. He is currently working as lecturer in cybersecurity program with La Trobe University, Australia. His research includes security, privacy and trust. He has published research work related to Data Sharing, Privacy, and Blockchain in different top research venues.