Received June 4, 2020, accepted June 19, 2020, date of publication June 30, 2020, date of current version July 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3006078

Performance Evaluation of Blockchain Systems: A Systematic Survey

CAIXIANG FAN[1], SARA GHAEMI[1], (Graduate Student Member, IEEE), HAMZEH KHAZAEI[2], (Member, IEEE), AND PETR MUSILEK[1,3], (Senior Member, IEEE)
[1]Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2R3, Canada
[2]Department of Electrical Engineering and Computer Science, York University, Toronto, ON M3J 1P3, Canada
[3]Department of Applied Cybernetics, University of Hradec Králové, 500 03 Hradec Králové, Czech Republic

Corresponding author: Caixiang Fan (caixiang@ualberta.ca)

This work was supported by the Future Energy Systems through the Canada First Research Excellence Fund (CFREF) at the University of Alberta.

ABSTRACT Blockchain has been envisioned to be a disruptive technology with potential for applications in various industries. As more and more different blockchain platforms have emerged, it is essential to assess their performance in different use cases and scenarios. In this paper, we conduct a systematic survey on the blockchain performance evaluation by categorizing all reviewed solutions into two general categories, namely, empirical analysis and analytical modelling. In the empirical analysis, we comparatively review the current empirical blockchain evaluation methodologies, including benchmarking, monitoring, experimental analysis and simulation. In analytical modelling, we investigate the stochastic models applied to performance evaluation of mainstream blockchain consensus algorithms. Through contrasting, comparison and grouping different methods together, we extract important criteria that can be used for selecting the most suitable evaluation technique for optimizing the performance of blockchain systems based on their identified bottlenecks. Finally, we conclude the survey by presenting a list of possible directions for future research.

INDEX TERMS Blockchain, distributed ledger technology, performance modelling, performance evaluation, systematic survey.

I. INTRODUCTION

Since its first introduction in Bitcoin by Nakamoto and Bitcoin [1] in 2008, blockchain has been recognized as a disruptive technology in various industries beyond cryptocurrency, including finance [2], [3], Internet of Things (IoT) [4], [5], health care [6], [7], energy [8]–[10] and logistics [11], [12]. Compared to conventional, centralized solutions, blockchain has some significant advantages such as immutability, enhanced security, fault tolerance and transparency. However, the decentralized nature of blockchain dramatically limits its performance (e.g., throughput and latency). For example, Bitcoin can only achieve a low throughput of 7 transactions per second (TPS), and it takes around 10 minutes for a transaction to get confirmed [13]. In contrast, current centralized payment systems such as VisaNet and MasterCard can reach thousands of TPS and almost real-time payments. By taking a similar consensus algorithm, proof-of-work (PoW), other blockchain platforms such as Ethereum [14] and Litecoin [15] also inherit the performance flaws of Bitcoin. Without doubt, the performance issue has become the major constraint of blockchain's applications in production. This is especially true for systems demanding high performance such as the online transaction processing (OLTP) and real-time payment systems.

To overcome this problem, many blockchains put efforts on improving their performance, e.g., by modifying the system structure and designing new consensus algorithms. These solutions include, but are not limited to, off-chain [16]–[19], side-chain [20]–[23], concurrent execution (smart contract) [24]–[26] sharding [27]–[31], and directed acyclic graph (DAG) [32]–[39].

Existing and new solutions should be comparatively evaluated in a meaningful manner to show their efficiency and effectiveness. For example, different versions of Hyperledger Fabric (HLF), e.g., HLF v0.6 and HLF v1.0, should be compared in the same evaluation framework to demonstrate the performance advantages/disadvantages of the new release.

The associate editor coordinating the review of this manuscript and approving it for publication was Ahmed Mohamed Ahmed Almradi.

In addition, through performance evaluation and analysis, bottlenecks can be identified and used to inspire further optimization ideas. Therefore, performance evaluation plays an important role in the area of blockchain research.

To this end, it would be useful to summarize, classify and survey the existing efforts on blockchain performance evaluation and to identify future directions in this area. However, most existing related surveys only focus on improvement (scalability) solutions or a specific evaluation topic of blockchain performance. A representative list of existing surveys, shown in Table 1, clearly identifies the need for a systematic survey on blockchain performance evaluation.

TABLE 1. Research scope of existing blockchain performance related surveys.

Year	Survey	Research scope
2018	Kim *et al.* [40]	scalability solutions
2019	Rouhani and Deters [41]	security, performance, and applications of smart contract
2019	Zheng *et al.* [42]	challenges of performance and security
2019	Wang *et al.* [43]	benchmarking tools and performance optimization methods
2020	Zhou *et al.* [44]	scaling solutions to blockchain
2020	Yu *et al.* [45]	sharding for blockchain scalability

In this contribution, we present a comprehensive, systematic survey on blockchain performance evaluation. The survey covers existing studies on evaluating the performance of various mainstream blockchains, and compares their advantages and disadvantages. It addresses the following research questions:

RQ1. What are the current mainstream techniques, main evaluation metrics and benchmark workloads for blockchain performance evaluation?

RQ2. How to comparatively evaluate the performance of two blockchain systems with different consensuses?

RQ3. What are the significant bottlenecks identified in various blockchain systems?

RQ4. What are the main challenges and opportunities in blockchain performance evaluation?

To answer these questions, the authors have searched and reviewed the latest papers published since 2015. The papers have been retrieved from major scientific databases, including *ACMDL, IEEEXplore, Elsevier, MPDI* and *SpringerLink*. In addition, closely related papers cited by the selected communications have also been taken into consideration. Note that this survey focuses only on blockchain performance evaluation, and solutions for blockchain performance or scalability improvement are not discussed. Interested readers may refer to the published surveys of performance improvement solutions for blockchain [40]–[45] listed in Table 1.

To the best knowledge of the authors, this is the first survey that systematically reviews the state-of-the-art on the blockchain performance evaluation from several different perspectives. The reviewed evaluation approaches can be classified into two high-level groups: empirical evaluation and analytical modelling, as shown in Figure 1. Empirical evaluation includes benchmarking, monitoring, experimental analysis and simulation. Analytical modelling mainly covers three types of stochastic models: Markov chains, queueing models and stochastic Petri nets (SPNs). Through this classification, we aim to depict a big picture of the performance evaluation landscape, identify current challenges in this area, and provide suggestions for future research. The contributions of this survey can be summarized as follows:

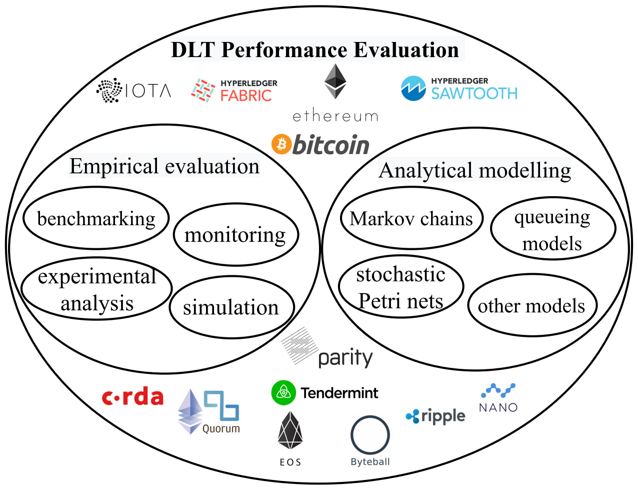


FIGURE 1. A landscape of DLT performance evaluation approaches and evaluated ledgers.

- It provides a systematic survey on the blockchain performance evaluation, covering all existing evaluation (empirical and analytical) approaches for evaluating the mainstream blockchain systems.
- It introduces existing popular models for analytical performance evaluation of prominent blockchain platforms, categorizes them and performs a comparative analysis of their advantages and disadvantages.
- It identifies the current challenges in this area, and subsequently provides suggestions for future research.

The remainder of this paper is organized as follows. Section II provides some prerequisite knowledge on distributed ledger technology (DLT), its categorization and architecture. Section III introduces the blockchain performance evaluation solutions from the perspective of empirical analysis, including benchmarking, monitoring, experimental analysis and simulation. Section IV focuses on the technical and mathematical introduction of existing commonly used performance modelling solutions including Markov chains, queueing models and stochastic Petri nets. The following Section V summarizes the major findings and points out potential opportunities in this area for future research according to the identified open issues. Finally, the survey is concluded in Section VI.

II. BACKGROUND

Blockchain is a major type of distributed ledger technologies (DLTs). The relationship of blockchain to DLT is just like the car to the vehicle [46]. As such, terms ''blockchain'' and ''DLT'' are used interchangeably throughout this paper. Any ledger that is stored in a distributed fashion and shared among a set of nodes or participants can be referred to as a distributed ledger. For new information to be added to this ledger, all participating nodes must reach a consensus on whether the information is legitimate or not. The algorithm which determines how this decision is reached, called *consensus algorithm*, is an important part of the DLT. In this section, we introduce a categorization of DLT and its abstraction layer architecture.

A. CATEGORIZATION OF DLTs

DLTs are widely used in cryptocurrencies such as Bitcoin [1], Ethereum [14], and EOS [47]. However, they can also be used in a variety of applications beyond cryptocurrencies. In 2019, *CB Insights* identified 55 industries that can be transformed by this technology [48].

A possible taxonomy of distributed ledger technologies is shown in Figure 2. DLTs can be categorized based on their data architecture. Two main categories are blockchain and directed acyclic graph (DAG). In blockchain, transactions are stored in containers called *blocks*, which are chained together using their hash values. This chain of information, similar to a linked list, is immutable. Examples of this category are Bitcoin, Ethereum, EOS, and Litecoin. In DAG, on the other hand, transactions are connected to one another by a reference relationship, forming a directed graph rather than a linked list. This category includes DLTs such as IOTA, Byteball, and Nano. In addition, there are distributed ledgers that have their unique data structure and do not fall into either of these categories, such as Radix and Corda.

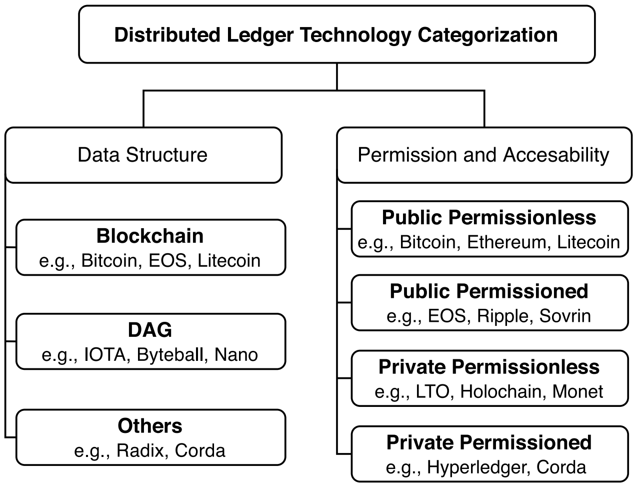

Distributed Ledger Technology Categorization
Data Structure
Permission and Accesability
Blockchain
e.g., Bitcoin, EOS, Litecoin
DAG
e.g., IOTA, Byteball, Nano
Others
e.g., Radix, Corda
Public Permissionless
e.g., Bitcoin, Ethereum, Litecoin
Public Permissioned
e.g., EOS, Ripple, Sovrin
Private Permissionless
e.g., LTO, Holochain, Monet
Private Permissioned
e.g., Hyperledger, Corda

FIGURE 2. Categories of distributed ledger technologies.

Based on the permissions of the ledger, DLTs can be classified as permissioned and permissionless, which usually makes one think of another taxonomy: private and public based on the ledger accessibility. In permissioned distributed ledgers, the identity of all the participants is known. By contrast, everyone can participate anonymously in a permissionless DLT network. Public and private DLTs can be distinguished by who can read the data on the ledger and verify its validity. Public ledgers are open and anyone can read the data on the ledger and host a node without the need to be approved. Private ledgers, by contrast, are only accessible by those who are pre-approved.

Therefore, based on the permissions and accessibility of the ledger, DLTs can be divided into four groups, as shown in Figure 2. Public permissionless ledgers, such as Bitcoin, Ethereum, and Litecoin, have no restriction on the participating parties. In public permissioned ledgers, the identity of participants should be known but anyone can read and validate the ledger. EOS, Ripple, and Sovrin are examples of this type. In private permissionless blockchains, the identities of the participants are not known but only pre-approved nodes validate the data. Examples of this category include LTO, Holochain, and Monet. Finally, in private permissioned ledgers, such as Hyperledger and Corda, access is restricted to pre-approved participants and the identities of the participants are known.

B. DLT ABSTRACTION LAYERS

Dinh *et al.* [49] introduced a blockchain design comprised of four identified abstraction layers, namely application, execution engine, data model and consensus. In the Oracle blockchain guidance book [46], the authors defined five layers to display the general architecture of blockchain, including the application and presentation layer, consensus layer, network layer, data layer and hardware/infrastructure layer. To better describe the architecture of DLT for the purpose of performance evaluation, we formulate an abstraction layer architecture following mainly Dinh's model [49], but extend it to five layers shown in Fig. 3.

1) APPLICATION LAYER

As the top presentation of DLT's technology stack, this layer contains the applications that are mainly used by the end users. Up to date, the most popular one is still cryptocurrency. As the first published digital currency, Bitcoin has controlled most of the marketplace and developed many variants. Ethereum has its own currency called *Ether*. IOTA also has its currency with the same name as the network, *IOTA* [37]. Other examples include the wallet to manage cryptocurrency, smart contracts, and all kinds of decentralized applications (*DApps*). In a DLT system, a smart contract is a piece of code designed to digitally facilitate, verify, or enforce the execution of a contract. For Ethereum, the smart contract is running on a dedicated virtual machine (called EVM); and most contracts

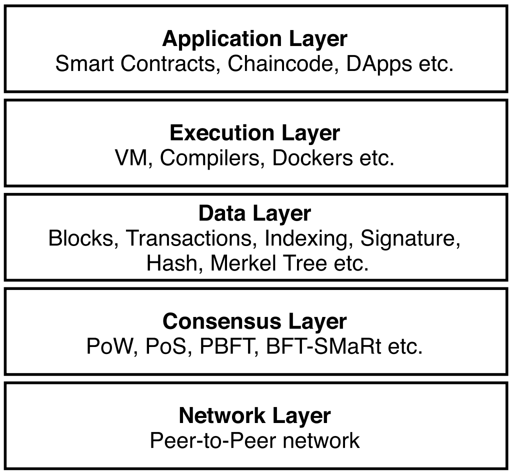

Application Layer
Smart Contracts, Chaincode, DApps etc.
Execution Layer
VM, Compilers, Dockers etc.
Data Layer
Blocks, Transactions, Indexing, Signature, Hash, Merkel Tree etc.
Consensus Layer
PoW, PoS, PBFT, BFT-SMaRt etc.
Network Layer
Peer-to-Peer network

FIGURE 3. Abstraction layer model for DLT.

on the system are related to cryptocurrency. While HLF's smart contract is running in a container such as Docker. One of the best-known DApps is the *decentralized autonomous organization (DAO)* in Ethereum, which creates communities to raise funding for exchange and investment.

Because this layer is in charge of presenting the final results executed from the distributed ledger system, it is supported and impacted by all the lower layers. Therefore, the performance evaluation results of the application layer reflect the overall performance of tested DLTs.

2) EXECUTION LAYER

The execution layer is in charge of executing contract or low-level machine code (bytecode) in a runtime environment installed on DLT network nodes. Ethereum has its own machine language and a virtual machine (EVM) developed to run the smart contracts code. Unlike Java virtual machine (JVM), the EVM reads and executes a low-level representation of smart contracts called the Ethereum *bytecode*. The smart contracts are programmed in a dedicated high-level language named *Solidity*, which is first compiled to bytecode by Solidity compiler. The Ethereum bytecode is an assembly language made up of multiple opcodes. Each *opcode* performs certain action on the Ethereum blockchain. In contrast, HLF does not take the semantics of language into consideration. It runs the compiled machine codes (from chaincode) inside Docker images. In addition, HLF's smart contract (chaincode) supports multiple general high-level programming languages such as Go, node.js, and Java rather than a dedicated language like Solidity of Ethereum. IOTA does not support smart contracts up to date. It adopts Java as the main development language and runs its reference implementation (IRI) in JVM. IOTA also has a version running in Docker image.

The runtime environment used to execute contracts or transactions needs to be efficient. And the execution result should be deterministic to avoid the uncertainty and inconsistency of transactions on all nodes. Any transaction abortion caused by inconsistent execution would result in computation resource waste and further decrease the performance. Additionally, the resource configurations (e.g., CPU and RAM) may impact the execution performance.

3) DATA LAYER

In the data layer, a wide range of data-related topics are defined, including transaction models, data structure, Merkel trees, hash function, encryption algorithms, etc. There are two popular transaction models: *unspent transaction output (UTXO)* and *account*. For UTXO, one owner completes value transfers by signing a transaction transferring the ownership of the UTXO to the receiver's public key. It involves an extra step of searching for ownership of the transaction from the sender's side. The account-based model is more efficient as it atomically updates two accounts in one transaction. A smart contract (chaincode for HLF) is a special type of account.

For blockchain, blocks containing transactions and contract execution states are chained together in a linked list by putting the hashed result of its previous block's content into the header of the current block. Ethereum and HLF employ a two-layer data structure to organize the block's content. All states are stored in a key-value database on a disk and indexed in a hash tree. The hash tree root is contained in the block's header. With a similar design, different DLTs have their own storage solutions for each level. For example, Ethereum uses LevelDB, and HLF uses CouchDB to store the states; Ethereum and Parity employ Patricia-Merkle (key-value store) tree, while HLF implements Bucket-Merkle tree to store the indices [49]. For IOTA, transactions are directly appended to the DAG structure called *tangle* in a hashed manner. The IRI uses RocksDB database to store the snapshot, a pruned ledger to prevent the tangle from expanding too large in size.

Besides the factors mentioned above, there are other design parameters in the data layer, such as hash functions (e.g., SHA 256 v.s. SHA 128), encryption algorithms (RSA v.s. ECC), and block size. All these factors may influence the performance of a blockchain system.

4) CONSENSUS LAYER

The consensus protocol is the core of a DLT system. It sets the rules and forces all nodes to follow them to reach an agreement (e.g., transaction confirmation) on blockchain content. Generally, there are two basic types of consensus mechanisms, which are proof-based and vote-based consensuses. The most popular proof-based consensus is proof-of-work (PoW), which has been employed by many blockchain systems. PoW is a very computation intensive consensus. It requires the nodes to solve a difficult puzzle to compete for the right of recording the ledger. Only the first node (called *winner*) solving the puzzle can append the block to the ledger and gets incentives accordingly. Since PoW provides high security, integrity and decentralization in an untrusted environment, it is very popular in public blockchains. However, the classic PoW protocol has a poor efficiency on processing transactions. To tackle this problem, many variations have been proposed to keep the same safety while achieving a better performance. Examples include greedy heaviest observed subtree (GHOST), proof of authority (PoA), proof of stake (PoS) and proof of elapsed time (PoET).

The vote-based consensuses are communication intensive. Different from PoW, vote-based solutions always give a deterministic execution result and usually achieve a relatively high performance. They rely on frequent message transitions among different roles in a network to ensure that all nodes reach an agreement on the block order. It is very popular in permissioned blockchains. Raft and Byzantine fault tolerance (BFT)-based, (e.g., PBFT and BFT-SMaRt) algorithms are two representatives of this consensus type. Raft has only crash fault tolerance (CFT), while PBFT and BFT-SMaRt can address Byzantine fault.

There are also some hybrid DLTs that combine different types of consensuses. For example, Tendermint combines PoS and PBFT; EOS takes a hybrid design combining PBFT and DPoS. Both target on improved performance and enhanced security. Interested readers may refer to the published surveys of blockchain consensus. Because of the deterministic properties, BFT-based consensus algorithms have a much lower transaction delay than PoW. But the expensive communication cost makes it difficult to scale, especially in a large network. Therefore, consensus design, evaluation and optimization in DLTs still remain an active research topic.

5) NETWORK LAYER

A peer-to-peer (P2P) network is the foundation of a DLT system. It takes care of peer discovery, transactions, and block propagation. In a public blockchain such as Bitcoin, this network is very large, with thousands of nodes working together to reach consensus. For private blockchain systems, the scale varies from several entities to over a hundred. Either way, a basic requirement for the P2P network is to provide speed and stablility. When a new participant wants to join, this layer ensures that nodes can discover each other. Then, all connected nodes communicate, propagate and synchronize with each other to maintain the current state of the blockchain network. Specifically, transaction broadcast, validation and transaction commit are all completed via this layer, as well as the world state propagation. In the P2P network, there are two basic types of nodes: full nodes and light nodes. Full nodes take care of mining, transaction validation and execution of consensus rules, while light nodes only keep the header of the blockchain (keys) and act as clients to issue transactions.

Therefore, the network layer is critical, especially for communication-intensive DLTs. Peer discovery and ledger synchronization among neighbours directly rely on the network, so that the speed determines the efficiency. And some detailed metrics, such as the number of transactions per network data are also related to this layer. Moreover, the package loss rate and network delay may have an impact on the performance of DLT.

III. EMPIRICAL ANALYSIS IN BLOCKCHAIN PERFORMANCE EVALUATION

In this section, we investigate existing approaches to blockchain performance evaluation from the perspective of empirical analysis. Specifically, different solutions, including benchmarking, monitoring measurements, self-designed experiments and simulation, are reviewed and compared. In practice, these approaches are usually used together to provide more evidence for blockchain performance evaluation.

A. BLOCKCHAIN BENCHMARKING TOOLS

The performance benchmarking has been well studied and documented for the cloud (e.g., Hadoop, Mapreduce and Spark) and database (e.g., relational and NoSQL) systems. Some proposed benchmark frameworks such as TPC-C [50], YCSB [51] and SmallBank [52] are well-established and have essentially formed the industrial standards. For example, YCSB is widely used for benchmarking NoSQL databases such as Cassandra [53], MongoDB [54] and HBase [55]; and SmallBank is a popular benchmark for OLTP workload.

However, these frameworks cannot be directly applied to benchmark distributed ledger systems due to the diversity of consensus mechanisms and APIs. As more and more blockchain systems emerge striving to improve DTL performance, it becomes imperative to devise a solution for comparing different platforms in a meaningful manner.

Up to date (June 2020), there are three popular performance benchmarks dedicated to evaluating blockchain systems, as listed in Table. 2.

Blockbench [49] is the first benchmark framework designed for evaluating private blockchains in terms of performance metrics on throughput, latency, scalability and fault-tolerance. Presently, it supports measurement on four major private blockchain platforms, namely Ethereum, Parity, HLF and Quorum. However, it claims to support the evaluation of any private blockchain by accordingly extending the workload and blockchain adaptors.

In the design of Blockbench, four abstraction layers in blockchain are identified: consensus, data model, execution engine and application, from the bottom (low level) to the top (high level). The consensus layer is in charge of setting the rule of agreement and getting all network participants to agree on the block content so that it can be appended to the blockchain. The data model defines the data structure, content and operations on the blockchain data. The execution engine contains resources of the runtime environment such as the EVM and Docker, which support the execution operations of blockchain codes. The application layer includes all kinds of blockchain applications such as smart contracts and different types of DApps. It is worth noting that Blockbench adopts and designs various workloads to test the performance of different layers, as shown in Table 3.

Hyperledger Caliper [57] is a performance evaluation framework mainly focusing on benchmarking Hyperledger blockchains such as Hyperledger Fabric, Sawtooth, Iroha,

TABLE 2. A comparison of three popular blockchain benchmarks.

Frameworks	Supported DLTs	Workloads Used	Evaluated Metrics	Pros & Cons
Blockbench [49]	Ethereum, Hyperledger Fabric (HLF), Parity and Quorum.	• macro: YCSB(k-v store), Smallbank(OLTP), EtherId, Doubler, and WavesPresale • micro: DoNothing, Analytics, IOHeavy, and CPUHeavy	throughput, latency, scalability and fault-tolerance.	adaptor-based framework, scalable; carefully designed workloads; but they are constant.
DAGbench [56]	IOTA, Nano and Byteball	• value/data transfer • transaction query: 1) input/output transaction numbers and 2) balance for a given account	throughput, latency, scalability, success indicator, resource consumption, transaction data size and transaction fee	adaptor-based framework, scalable; specific for DAG DLT; workloads are not representatives.
Hyperledger Caliper [57]	Hyperledger blockchains (Fabric, Sawtooth, Iroha, Burrow and Besu), Ethereum, FISCO BCOS	Self-defined in the configuration file	throughput, latency, resource consumption, success rate	adaptor-based framework, scalable; no pre-defined workload design, but support more DLT systems.

TABLE 3. Blockbench workloads for evaluating each layer of blockchain.

Layer	Benchmark Workload		Workload Description	Measurement Identifier
Application	Macro Workloads	YCSB	Key-value store	throughput and latency
		Smallbank	OLTP workload	throughput and latency
		EtherId	Name registrar contract	throughput and latency
		Doubler	Ponzi (pyramid) scheme	throughput and latency
		WavesPresale	Crowd sale	throughput and latency
Execution Engine	Micro Workloads	CPUHeavy	Sort a large array	latency
Data Model		VersionKVStore	Keep state's versions (HLF only)	latency
		IOHeavy	Read and write a lot of data	latency
Consensus		DoNothing	Simple contract, do nothing	latency

Burrow and Besu. In the system architecture, there are two main components: Caliper core and Caliper adaptors. The former defines system workflow, while the latter are used to extend evaluation for other blockchains such as Ethereum and FISCO BCOS. Before running a test, benchmark workloads and necessary information interfacing adaptor to the system under test (SUT) need to be predefined in configuration files. During the test, a resource monitor runs to collect resource utilization information (e.g., CPU, RAM, network and IO) and all clients publish their transaction rate control information to a performance analyzer. When a test is finished, a detailed test report is generated by a report generator.

DAGbench [56] is a relatively recent framework dedicated to benchmarking DAG distributed ledgers such as IOTA, Nano and Byteball. The currently supported indicators are throughput, latency, scalability, success indicator, resource consumption, transaction data size and transaction fee. From the system design perspective, DAGbench shares the same approach with Blockbench and Caliper which adopt a modular adaptor-based architecture. Users just need to choose (or develop if not available) associated adaptors for different workloads and blockchain systems under evaluation.

Besides the general performance metrics evaluation, there are also studies focusing on specific metrics for particular blockchain. For example, OpBench [58] and another benchmark framework [59] are proposed to evaluate if a miner's award is proportional against to the CPU execution time or consumed computation power for Ethereum smart contracts.

B. BLOCKCHAIN PERFORMANCE MONITORING

Blockchain benchmarking usually requires a standardized environment and a well-documented workload as input. However, for public blockchain systems, it is impossible to have a good control against the real workload and consensus participants, which makes the benchmarking more challenging. In terms of evaluating public blockchains, there are two potential solutions.

The first solution is to build a private version of the associated test network and leverage the existing benchmarks mentioned above to evaluate blockchain under artificially designed workloads. This may require new adapter development for either workload or blockchain network. In addition, this approach should take into consideration the scalability problem of blockchain because the tested private version of blockchain may encounter scaling issues when implemented publicly. Therefore, the tested result may show better values of performance metrics compared to the real public network.

The second solution is to monitor and evaluate the live public system's performance under realistic workload [60]. Zheng *et al.* [61] proposed a detailed, real-time performance monitoring framework using a log-based approach. It has lower overhead, more details, and better scalability compared to its counterpart solution via remote procedure call (RPC). The high-level system framework is shown in Fig. 4.

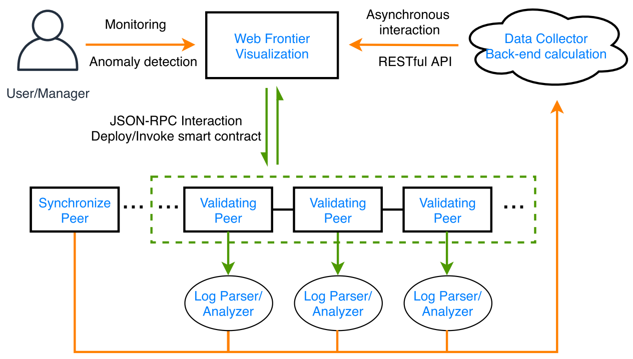


FIGURE 4. Blockchain performance monitoring framework [61].

C. EXPERIMENTAL ANALYSIS OF BLOCKCHAIN SYSTEMS

In this section, we look at DLT performance evaluation from the perspective of empirical analysis based on self-designed experiments. Even though empirical analysis can hardly provide standardized test results like benchmarking, this approach is very flexible in parameterization. It can be used to identify potential bottlenecks and pave the way to further performance improvements.

Experiment-based approaches have been widely employed to evaluate distributed ledger systems such as Hyperledger, Ethereum and DAG-based ledger. Various private blockchain platforms and different versions of a certain blockchain can be compared on performance by running tests under a well-controlled test environment. In addition, some studies examined the detailed performance, for example, the performance of different encryption and hash algorithms, from the data layer in the blockchain abstraction model.

1) HYPERLEDGER PERFORMANCE ANALYSIS

Nasir *et al.* [62] conducted an experimental performance analysis of two versions of HLF (v0.6 and v1.0) on their execution time, latency, throughput and scalability by varying the workloads and node scales. The overall results indicate that HLF v1.0 consistently outperforms HLF v0.6 across all evaluated performance metrics.

Baliga *et al.* [63] took an experimental approach to study throughput and latency of HLF v1.0. Using Caliper as the benchmarking tool, the authors configured different transaction and chaincode parameters to explore how they impact transaction latency and throughput under micro-workloads. Fabric's performance characteristics were also studied by varying the number of chaincodes, channels and peers. The results show that the throughput of HLF v1.0 is sensitive to the orderer settings, and it is a significant drawback for the commiter in this version that it does not process transactions in parallel, incapable of taking advantage of multiple vCPUs.

Another comprehensive empirical study was conducted by Thakkar *et al.* [64] who 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 vs. CouchDB). The experimental results indicated that endorsement policy verification, sequential policy validation of transactions in a block, and state validation and commit (with CouchDB) were the three major bottlenecks. Accordingly, the authors suggested three optimization solutions, including parallel VSCC validation, cache for membership service provider (MSP), and bulk read/write for CouchDB. All these optimizations have been implemented in release HFL v1.1.

A study completed at IBM by Androulaki *et al.* [65] focused on HLF v1.1 to explore the impact of block size, peer CPU, and SSD vs. RAM disk on blockchain latency, throughput and network scalability under different numbers of peers. The results show that HLF v1.1 achieves end-to-end throughput of 3500+ TPS in certain popular deployment configurations, with the latency of a few hundred *ms*, scaling well to 100+ peers.

Nguyen *et al.* [66] conducted an experimental study to explore the impact of large network delays on the performance of Fabric by deploying HLF v1.2.1 over an area network between France and Germany. The results reveal that an obvious network delay (3.5s) brings 134 seconds offset after the 100th block between two clouds, which indicates that the tested version of Fabric can not provide sufficient consistency guaranties. Therefore, HLF v1.2.1 cannot be used in critical environments such as banking or trading. This was the first work that experimentally demonstrated the negative impact of network delays on a PBFT-based blockchain.

Another HLF performance evaluation work focusing on the underlying communication network was conducted by Geyer *et al.* [67] using Caliper [57] and a dedicated testbed on which network parameters, such as latency or packet loss, can be configured. In the experiments, the influence of transaction rate, chaincode, network properties, local network impairment, and block size have been separately examined and quantitatively analyzed. The experiment results identified the validation of the transactions as the major contributor to the transaction latency in HLF.

As the first long-term support release, HLF v1.4 caught the attention of several blockchain researchers. Kuzlu *et al.* [68] investigated the impact of network workloads on the performance of a blockchain platform in terms of transaction throughput, latency, and scalability (i.e., the number of participants serviceable by the platform). Following network load parameters were varied in the experiment: number of transactions, transaction rate and transaction type.

Although the practical Byzantine fault tolerance (PBFT) algorithm has been adopted as the consensus protocol since its version 0.6, dishonest participants and their attacks such as intentionally delaying messages, sending inconsistent messages and distributed-denial-of-service (DDoS) never stop. Malicious behaviour may significantly undermine the system in terms of both security and performance. To explore the performance of HLF with malicious behaviour, Wang [69] designed multiple malicious behaviour patterns and experimentally tested the transaction throughput and latency on HLF. The results show that delay attacks, along with keeping some replicas out of working, dramatically decrease the system performance.

Apart from Fabric, Shi *et al.* [70] empirically studied the performance of Sawtooth, another well-known permissioned blockchain platform from Hyperledger. The examined performance metrics include consistency (i.e., whether the platform's performance behaves consistently each time with the same workload and cloud VM configuration), stability (i.e., whether the platform's performance remains stable with the same workload, but different cloud VM configurations) and scalability (i.e., how the platform performance achieves scalability with different workloads and configuration parameters). The adjustable configuration parameters identified for optimizing the performance of Sawtooth are scheduler and maximum batches per block.

From the results of empirical performance analysis summarized above, it is obvious that Hyperledger needs improvement on both *geographical scalability* (limited by the network latency) [66] and *size scalability* (the platform fails scaling beyond 16 nodes [49]). The bottleneck is the adopted PBFT consensus, which is a communication bound mechanism as opposed to the computation intensive PoW [71] consensus.

2) ETHEREUM PERFORMANCE ANALYSIS

Rouhani and Deters [72] studied the performance of Ethereum on a private blockchain by analyzing two most popular Ethereum clients: PoW-based Geth and PoA-based Parity. The results indicate that, compared to Geth, Parity is 89.82% faster in terms of transaction processing, on average, under different workloads.

Yasaweerasinghelage *et al.* [73] introduced an approach to predict the latency of blockchain-based systems using software architectural modelling tool Palladio workbench [74] and simulation. They leveraged the proposed method to test latency on a private Ethereum (Geth) experimental environment. The results show a low relative error of response time, mostly under 10%.

Bez *et al.* [75] conducted an initial quantitative analysis on the scalability of Ethereum. The transaction throughput was evaluated under an extensible test environment with synthetic benchmarks. The results indicate that Ethereum follows the scalability trilemma, which claims that a blockchain platform can hardly reach decentralization, scalability and security simultaneously.

3) DAG DLT PERFORMANCE ANALYSIS

In traditional blockchain systems, transactions are stored in blocks that are then organized as a ledger in a single chain data structure. This structure makes it incapable of concurrently generating blocks, and thus limiting the transaction throughput. In DAG-based distributed ledgers, transactions or blocks are organized in different vertices of the directed graph, which allows parallel block generation and inclusion. Based on this idea, many distributed ledgers have been proposed with their own consensus mechanisms. For example, IOTA [37] employs a cumulative weight approach for transaction confirmation and Markov chain Monte Carlo (*MCMC*) sampling algorithm for random *tip selection*; Byteball [38] achieves consensus by relying on 12 selected reputable *Witnesses*; and Nano [39] adopts a balance-weighted vote mechanism to reach agreement on transaction confirmation.

Even though DAG-based ledgers are designed to theoretically have faster transaction speed than blockchain, it is necessary to evaluate the performance of existing DAG distributed ledger implementations and identify their potential bottlenecks. Fan *et al.* [76] demonstrated the scalability of IOTA under IoT scenarios in a private network with 40 nodes. The experiment results indicated that transaction throughput (TPS) has good linear scalability against the transaction arrival rate. Three representatives of DAG-based distributed ledgers, namely IOTA, Nano and Byteball, were comparatively evaluated using the proposed DAGbench in [56]. From the experimental results, some useful observations, such as the advantages and disadvantages of the three tested DAG implementations, can be obtained.

4) COMPARATIVE ANALYSIS

Before developing a blockchain-enabled application, decision makers should first assess the suitability of blockchain implementation. Then, a comparative performance analysis is often necessary to select a blockchain platform that will perform well in the target application environment.

After developing Blockbench, Dinh *et al.* [49] used this tool to conduct a comparative performance analysis on three mainstream private blockchains, namely Ethereum (geth v1.4.18), Parity (v1.6.0) and HLF (v0.6.0-preview). Their findings can be summarized as follows: 1) HLF performs consistently better than Ethereum and Parity across all macro (e.g., throughput and latency) and micro (e.g., IOHeavy) benchmarks, but it fails to scale up to more than 16 nodes; 2) The consensus protocols are identified as major bottlenecks for HLF and Ethereum, while transaction signing is a bottleneck for Parity. The authors further compared the performance of two different versions of HLF v0.6.0 and v1.0.0 against IOHeavy workload in their more recent work [71].

Because of the lack of interface standards, evaluating different blockchains remains difficult. To overcome this problem, a generic workload performing the same functions on different blockchain interfaces was designed in [77] to comparatively evaluate three prominent consortium blockchain platforms for IoT. They were HLF v0.6 with the PBFT consensus, HLF v1.0 with the Byzantine fault-tolerant state machine replication (BFT-SMaRt) consensus, and Ripple with the Ripple consensus. Results confirmed that the evaluated blockchains could offer reasonable throughput but with very limited scalability.

Pongnumkul *et al.* [78] conducted a preliminary performance analysis of two popular private blockchain platforms HLF (v0.6) and Ethereum (geth 1.5.8, private deployment) under varying workloads. The experimental results demonstrated that HLF outperforms Ethereum in terms of all evaluated metrics (execution time, latency and throughput). However, this study also pointed out that the performances of both platforms are still not competitive with current mainstream database systems, especially under high workloads. This conclusion was tested and confirmed by another, more recent study [79], in which Ethereum and MySQL were compared.

Comparative analysis can also be conducted on consensus algorithms of different blockchains. For example, Hao *et al.* [80] compared the performance between Hyperledger (PBFT) and private Ethereum (PoW) via their proposed benchmark framework constructed with four modules: workload configuration module, consensus smart contract module, data collector module and the target blockchain platforms. The evaluation results show that HLF consistently outperforms Ethereum in terms of average throughput (TPS) and latency. This study also points out that the consensus mechanism induces performance bottleneck in private blockchains. Another example is the performance analysis conducted on PoW and the Proof-of-Collatz Conjecture (PCC) [81]. PCC [82] is a recently introduced number-based theoretic PoW using a new metric called *Collatz orbits*, which are defined in the Collatz Conjecture algorithm. Authors evaluated these two consensus algorithms with respect to the execution time, the deployment time and the latency on a private blockchain network. The experiment results demonstrate that PCC-based blockchain consistently outperforms PoW-based blockchain in terms of all tested metrics and even steadily achieves 1000× faster execution speed than of PoW.

To provide system designers suggestions on smart contract platform selection, Benahmed *et al.* [83] conducted a comparative performance analysis of Hyperledger Sawtooth, EOS and Ethereum. Following the workloads used in Blockbench [49], the authors modified and defined three types of workloads, namely CPUHeavy, KVStore (Key-Value Store), and SmallBank, to comparatively test CPU consumption, memory consumption, load scalability and network scalability in distributed ledgers. The results reveal that the third generation platform EOS outperforms the other two in both resource consumption and speed, but with some shortcomings such as centralization. In addition, according to their performance in the test, Sawtooth was suggested for use in the Internet of Things and Ethereum's PoA implementation for the fast development of web-oriented applications.

To explore whether existing blockchain solutions can scale to large IoT networks, Han *et al.* [84] comparatively evaluated the performance of five selected prominent distributed ledgers using classic consensus protocols: Ripple, Tendermint, Corda and v0.6 and v1.0 of HLF. A series of exclusive tests were run to evaluate the throughput and latency with different numbers of nodes (ranging from 2 to 32) for each of the ledgers. The results show that even though these systems can sometimes provide thousands of TPS throughput, their networks usually do not scale to tens of devices as the performance drops dramatically when the number of nodes increases. Table 4 lists an overview of various DLTs' performance extracted from the reviewed experimental analysis studies.

5) ENCRYPTION PERFORMANCE ANALYSIS

In addition to the end-to-end performance metrics, there are also some evaluation works focusing on the detailed performance of a certain step or subprocess such as the efficiency of encryption and hash function. According to Park *et al.* [85], the transaction processing time equation is

$$T = t_i + t_c = (t_v + t_{pow} + t_n + t_e) + t_c, \tag{1}$$

where t_i refers to the issuance time, t_c to the confirmation time, t_v to the validation time, t_{pow} to the PoW time, t_n to the network overhead, and t_e to the processing overheads. The processing overheads include encryption/decryption, hashing and authentication. Efficient encryption and hashing algorithms contribute to transaction issuance speed in DLT. Chandel *et al.* [86] analyzed and compared the performance of the two most commonly used encryption algorithms in blockchain, Rivest-Shamir-Adleman (RSA) and elliptic-curve cryptography (ECC). Their comprehensive analysis results based on the key size, key generation performance and signature verification performance show that the ECC algorithm (adopted by Bitcoin and Ethereum) outperforms RSA in general. This study also points out that ECC satisfies the security needs of blockchain better than RSA.

More recently, Ferreira *et al.* [87] conducted a study on Blockchain-based IoT (BIoT) [88] to explore the performance of hash function in blockchains. Particularly, authors developed a blockchain in an IoT scenario to evaluate the performance of different cryptographic hash functions such as MD5, SHA-1, SHA-224, SHA-384 and SHA-512. The test results show that SHA-224 and SHA-384 are the best hash functions for blockchain due to their lack of collision attacks. In hashing ciphers, a collision attack is the problem that there exist two different messages m_1 and m_2, such that $\text{hash}(m_1) = \text{hash}(m_2)$. In addition, these two hash functions are more time-efficient than others to process blockchain functions with the advantage of producing a smaller average block size.

TABLE 4. Overview of different DLT performance (throughput and latency) under various evaluation environments.

DLT	Consensus	Throughput (TPS)	Latency (Secs)	Workload	Network (Size)	Node Configuration
HLF v0.6 [49]	PBFT	1273	38	YCSB	8 nodes	E5-1650 3.5GHz CPU, 32GB RAM, 2TB HD
	PBFT	1122	51	Smallbank	8 nodes	
Ethereum geth v1.4.18 [49]	PoW	284	92	YCSB	8 nodes	
	PoW	255	114	Smallbank	8 nodes	
Parity v1.6.0 [49]	PoA	45	3	YCSB	8 nodes	
	PoA	46	4	Smallbank	8 nodes	
Quorum 2.0 [63]	Raft	2,000+	1.5	write-only/null	3 nodes	8 vCPUs 4 cores 3.6 GHz, 16GB RAM
	IBFT	1,900	3.2	null	4 nodes	
	IBFT	1,800	3.5	write-only	4 nodes	
HL Sawtooth v1.1.2 [83]	Proof of Elapsed Time (PoET)	3	-	Smallbank	6 nodes	Dockers share VM on Intel
EOS v1.5.3 [83]	Delegated Proof of Stake (DPoS)	21	-	Smallbank	6 nodes	Xeon X7350 CPU 16 Cores,
Ethereum Geth v1.8.21 [83]	PoW	10	-	Smallbank	6 nodes	2.93GHz, 64GB RAM
HLF v1.0 [77]	BFT-SMaRt	1,700	-	Payment transaction	16 nodes	E5-2630 CPU
HLF v0.6 [77]	PBFT	2600	1.8	Invoking chaincode	16 nodes	8 cores 2.4GHz,
Ripple v0.60.0 [77]	XRP	1450	6	Payment transaction	16 nodes	64GB RAM
Tendermint v0.22.4 [84]	PBFT and Casper	6,000	0.15	Invoke Payment transaction	16 nodes	E5-2630 CPU
Tendermint v0.22.4 [84]	PBFT and Casper	5,600	0.05	Query Payment transaction	16 nodes	4 cores 2.4GHz,
R3 Corda v3.2 [84]	BFT-SMaRt	50	8	Query Payment transaction	4 nodes	12GB RAM
Geth v1.7.3 [80]	PoW	130	1,297	YCSB(N=10,000)	4 nodes	
Geth v1.7.3 [80]	-	235	569	YCSB(N=10,000)	4 nodes	8GB RAM,
HLF v1.0 [80]	BFT-SMaRt	535	78	YCSB(N=10,000)	4 nodes	128GB SSD
HLF v1.0 [80]	-	1,033	40	YCSB(N=10,000)	4 nodes	
Geth [72]	PoW	-	0.199	Payment transaction	1 node	Core i7-6700
Parity [72]	PoW	-	0.105	Payment transaction	1 node	CPU, 24GB RAM
Geth 1.5.8 [78]	-	21	361	TransferMoney (N=10,000)	1 node	AWS EC2 Intel E5-1650
HLF v0.6 [78]	-	160	4	TransferMoney (N=10,000)	1 node	8 core CPU, 15GB RAM, 128GB SSD

D. SIMULATION

All the evaluation solutions mentioned above (i.e., benchmarking, monitoring and experimental analysis) require the availability of the systems, no matter private or public blockchains. However, the system under evaluation is not always available. For instance, when a company needs to make a selection between two blockchain platforms under development according to their performance, none of the previously discussed solutions is feasible. Moreover, it is usually costly on both time and resources to construct a real blockchain network for testing. This brings us to explore another evaluation approach, namely, *simulation*. A blockchain simulator can mimic the behaviours of network nodes in reaching the consensus, providing performance similar to a real system. Besides, a blockchain simulator usually provides a convenient way for users to tune the system parameters to run different settings for the sake of comparison. In this subsection, we will take a look at the role of simulation in the blockchain world.

1) BlockSIM

In 2019, there were three similar simulators with the same name, *BlockSim* (or *BlockSIM*), proposed for simulating blockchain systems. Alharby and van Moorsel [89] proposed and implemented a framework called BlockSim to build discrete-event dynamic system models for PoW-based blockchain systems. This framework was organized in three layers: incentive layer, connector layer and system layer. Using the proposed simulation tool, the authors explored the block creation performance under the PoW consensus

TABLE 5. A comparison on different empirical performance evaluation solutions for blockchain system.

Solutions	Characteristics			Efficiency		
	Node	Network	Workload	Parameterization	Extensibility	Difficulty
Monitoring	Real	Real	Real	Low	Low	Easy
Benchmarking	Real	Test	Artificial	Low	High	Easy
Experimental Analysis	Real	Test	Artificial	High	Low	Medium
Simulation	Virtual	Virtual	Virtual	Very High	Very High	Hard

algorithm. This simulator helped to understand the details of the block generation process in PoW. The predefined test cases were validated and verified in their extension study, where the simulation outcomes were compared with results of real-life systems such as Bitcoin and Ethereum to show the feasibility of this approach. However, the extensibility of this simulator is still a problem for future research.

To help architects better understand, evaluate and plan for the system performance, Pandey *et al.* [90] proposed and developed a comprehensive open-source simulation tool called BlockSIM for simulating private blockchain systems. This tool is designed to evaluate system stability and transaction throughput (TPS) for private blockchain networks by running scenarios, and then decide on the optimal system parameters suited for the purposes of architects. The comparison results between BlockSIM and a real-world Ethereum private network running PoA consensus show that BlockSIM can be used effectively.

More recently, Faria and Correia [91] presented a flexible discrete-event simulator (also called BlockSim) to evaluate different blockchain implementations. With a good design of APIs, new blockchains can be easily modelled and simulated by extending the models. Running this simulator for Bitcoin and Ethereum, the authors got some interesting performance conclusions. For instance, doubling the block size (number of transactions) had a small impact on the block propagation delay (10ms), while encrypting communication had a higher impact on that delay (more than 25%).

2) DAGsim

Similarly, Zander *et al.* [92] presented a continuous-time, multi-agent simulation framework called *DAGsim*, for DAG-based distributed ledgers. Specifically, the performance of IOTA in terms of the transaction attachment probability was analyzed using this tool. The results indicate that agents with low latency and high connection degrees have a higher probability of having their transactions accepted in the network. Another multi-agent tangle simulator [93] built with NetLogo simulates both random uniform and MCMC tip selection in a visualized and interactive way.

In addition to pure simulators, some other studies leverage simulations combined with analytical results to conduct validation or exploration. Park *et al.* [85] proposed and implemented a general DAG-based cryptocurrency simulator using Python. This simulator was used to validate the proposed analytical performance model, through which they found that by issuing a transaction with a smaller average number of parents n in DAG, the transaction speed (TPS) can be increased. Kusmierz *et al.* [94] ran IOTA tangle simulations in a continuous-time model to explore how different tip selection algorithms, i.e., uniform random tip selection (URTS) and unbiased random walk (URW), affect the growth of the tangle. Simulations under varying transaction arrival rates were used to analyze the performance of the tangle.

E. COMPARISON OF DIFFERENT EVALUATION SOLUTIONS

In the previous subsections, we introduced four types of empirical evaluation solutions and surveyed existing studies which adopted the associated approaches. In this subsection, we comparatively discuss the advantages and disadvantages of the above-mentioned solutions. This comparison is based on both the general characteristics of the individual approaches and their suitability in evaluating different types of blockchains. The compared items are divided into two categories: solution requirements and solution efficiency, see Table 5. Solution requirements describe the network specifications for evaluating blockchain systems in terms of the node, network and workload. Solution efficiency provides three dimensions, namely parameterization, extensibility and difficulty, to compare the efficiency and effectiveness of different solutions.

Monitoring the performance of a blockchain system requires a realistic deployment of the system in production with real workloads. Even though this approach can also be used to evaluate a private blockchain in an experimental setup, we argue that it is more suitable to evaluate public blockchain when compared with benchmarking. In the context of evaluating a public blockchain, it becomes difficult to change any parameters for multiple tests. The challenge of the extensibility lies in the development of adaptable log parser for various blockchains. But, it is easy to deploy for certain blockchains using the existing solutions [61].

Benchmarking requires a well-controlled evaluation environment with a test network and artificial workloads. Once a benchmark tool is selected, the supported workloads and test metrics are limited, as well as the parameters which can be tuned. For example, Blockbench doesn't support tuning the network layer parameters such as network delay and, up to date, it only supports evaluating four types of blockchain platforms, i.e., Ethereum, HyperledgerFabric (HLF), Parity and Quorum. However, the well-designed APIs allow users to develop their own adaptors and extend its feasibility to evaluate any private blockchains. So, the extensibility of benchmarking is relatively higher than monitoring. In addition, this solution is easy to deploy since there have been several popular and well-documented benchmark tools, see Table 2.

Experimental analysis refers to the evaluation solution based on self-designed experiments. This is a very general solution that is commonly used. It is very similar to benchmarking but different in two main aspects. First, self-design gives more flexibility in evaluating impact factors, providing a high capability of parameterization. For example, the impact of network delay on HLF performance can be evaluated in a self-designed experiment, which is not supported by benchmarking. Second, the test is usually dedicated to a specific blockchain and is not as standardized as benchmarking, which limits the extensibility. So, the deployment difficulty partly depends on the complexity of the SUT and what to evaluate.

Simulations have a relatively greater difficulty in the stage of simulator design and development. But, once it is completed, the simulator usually provides a number of advantages in comparison to other approaches. The simulation solution is very extensible and can be used to quickly test different configuration parameters at a low cost. As mentioned in subsection III-D, another obvious advantage of simulation is that it does not require the availability of the blockchain. However, as for the evaluation results, there may be a relatively large difference (e.g., 10%) between simulation and experiment, which induces concerns about the accuracy of this solution. Moreover, some metrics cannot be evaluated in simulators such as the transactions per CPU, transactions per memory second, transactions per disk IO, and transactions per network data for a blockchain system.

IV. ANALYTICAL MODELLING IN BLOCKCHAIN PERFORMANCE EVALUATION

Analytical modelling of performance leverages mathematical tools to formalize blockchain system in an abstract way and to solve ensuing models with rigor. The model output (e.g., average transaction latency being expressed as a function of network indicators) provides analytical evidence for blockchain performance evaluation. In this section, we survey the performance evaluation solutions of distributed ledger systems based on analytical modelling. We aim to summarize the mainstream techniques, explore how and why these models are employed for certain distributed ledgers, and then identify the current challenges in blockchain performance modelling. In particular, we focus on surveying the stochastic models, which have been used to successfully model many cloud systems.

In Table. 6, we classify the existing popular solutions of performance modelling for distributed ledgers into four categories: Markov chains, queueing models, stochastic Petri nets and other models.

A. MARKOV CHAINS FOR MODELLING DLT CONSENSUSES

In probability theory, Markov processes are a type of stochastic process with *Markov property*. Also called *memoryless property*, it refers to the fact that the future states of the process depend *only* on the present state, but not on the previous ones. *Markov chain* is defined as a Markov process with discrete state space. It is a fundamental mathematical tool to evaluate the performance of distributed ledger systems [111]. In this subsection, we investigate how Markov chains are used to model two different consensus algorithms: *Raft* and *the tangle* for IOTA. The specific type of process used for this modelling is called discrete time Markov chain (DTMC).

DTMC for Modelling Raft: In a Raft [112] cluster, each node is at any given time in one of the three states: follower, candidate and leader. Normally, there is only one leader in a Raft cluster. We call it *network split* in the case of two or more leaders being elected simultaneously, which may dramatically impact the performance of the system. After the leader has been elected, it handles all requests from the client and sends them to followers for validation. Followers simply receive requests from and respond to leaders and candidates. Candidates are a mid-state transiting from follower to leader. The whole Raft consensus can be divided into several ever increasing timely manners called *terms* which have two processes: leader election and ledger replication. Each term starts with a leader election, in which all nodes start from follower state. Then, a node transits to candidate, candidate to leader or back to follower according to the rules depicted in Fig. 5 [112]. Once a leader is elected successfully, the ledger replication process starts with the leader sending heartbeat messages to all other nodes to establish its authority and prevent new elections. Once the leader receives responses of writing new transaction entry to the ledger from the majority of the followers, it notifies them and the client that the transaction is committed.

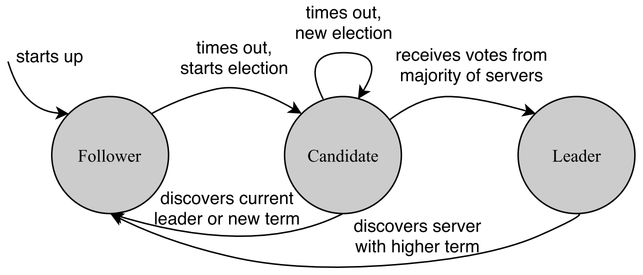


FIGURE 5. Node states transition illustration in Raft consensus.

To explore the impact of network properties on the blockchain performance, Huang *et al.* [95] have built a simple Markov chain model for the process of a node transferring from follower state to candidate. They consequently present the network split probability as a function of the network size, the packet loss rate, and the election timeout period. Let us define the packet loss probability as a constant value p for a given network, the timeout period for each round of election as E_t uniformly initiated from a range $[a,b]$, and interval

TABLE 6. A summary of blockchain performance modelling studies.

Model Types	Models	Consensus	DLs	Model Outputs
Markov chains	Absorbing Discrete Time Markov chain (DTMC) [95]	Raft	private blockchains	network split probability as a function of packet loss rate, election timeout, and network size
	Discrete Time Markov chain (DTMC) [96]	the tangle	IOTA	cumulative weight and transaction confirmation delay
queueing models	$M/G^B/1$ queue variant [97]	PoW	Bitcoin	tx confirmation time
	CTMC GI/M/1 queue [98]	PoW	Bitcoin	mean number of txs in the queue and in a block; average tx-confirmation time.
	CTMC GI/M/1 queue [99]	PoW	Bitcoin	mean stationary number of txs in the queue and in the block
	M/G/1 queue variant [100]	PoW	Bitcoin	confirmation time and tx delay
	non-exhaustive queue [101]	PoW	NA	mean number of txs and mean confirmation time of txs in the system
	Discrete-time $GI/GI^N/1$ queue [102]	Proof-of-Authority	Ethereum	system queue size and tx waiting time
	$M/M^B/1$ queue [67]	BFT-SMaRt	HLF	tx latency
	M/G/1 and M/M/1 queue [103]	PBFT	NA	system delay
	(n,k) fork-join queue [104]	vote-based consensus	permissioned blockchain	blocks commitment delay, block validation response time and synchronization processes among mining nodes.
	Fluid queue [105]	the tangle	IOTA	conflicting txs cannot coexist when a random tip-selection algorithm is employed
stochastic Petri nets	Generalized stochastic Petri nets (GSPN) [106]	PBFT	HLF v1.2	latency and throughput of each phase
	Stochastic Reward Nets(SRN) [107]	PBFT	HLF v1.0	throughput, utilization and mean queue length at each peer
other models	World State Prediction model [108]	PoW	Ethereum	transaction time cost
	Stochastic network model [109]	PoW	Ethereum	tx processing rate
	Random Graph model [110]	PoW	Bitcoin	block propagation delay and traffic overhead

between two heartbeats as τ. Thus, the maximum number of heartbeats for an election to timeout is $K \in \{K_1, K_2, \ldots, K_r\}$, where $K_1 = \lfloor a/\tau \rfloor$ and $K_r = \lfloor b/\tau \rfloor$. Then, two discrete time stochastic processes at time n can be defined: $\boldsymbol{g(n)}$ as the stage status $\{1,2,\ldots,r\}$ of a given node, and $\boldsymbol{b(n)}$ as the remaining steps (i.e., number of heartbeats) for the election phase to timeout in a term.

Therefore, the transition process for an observed node from follower to candidate can be modelled as a two-dimensional stochastic process $\{\boldsymbol{g(n)},\boldsymbol{b(n)}\}$. It can be further transformed to an absorbing DTMC on the state space $\{(1,K_1),\ldots,(1,0),\ldots,(i,K_1),\ldots,(i,0),\ldots,(r,K_r),\ldots,(r,0)\}$.

Using the mathematical derivations proven in [95], the network split probability before n-th step can be obtained.

DTMC for Modelling IOTA Tangle: IOTA tangle [37] is a DAG-based distributed ledger designed for the microtransactions in the IoT. Its consensus encourages all participants to contribute in maintaining the ledger through referencing (i.e., approving) two unapproved transactions called *tips* before issuing any new transaction. For the new coming transaction, IOTA tangle leverages the MCMC random walk algorithm to select two tips. All transactions directly or indirectly approved by this new transaction then add its weight to their cumulative weights, as shown in Fig. 6. For an approved transaction, its cumulative weight gradually increases to reach a predefined threshold. Finally, the corresponding transaction is considered confirmed and permanently recorded in the ledger.

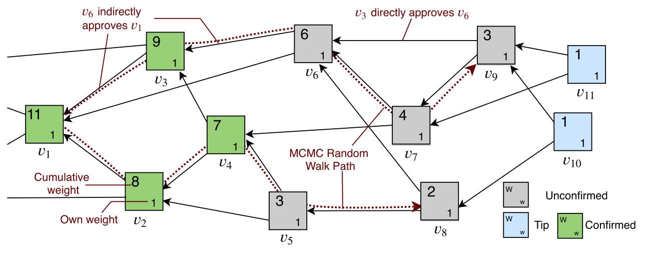


FIGURE 6. An example of the IOTA tangle.

To explore the impact of various transaction arrival rates on the cumulative weight and confirmation delay of an observed transaction, Cao *et al.* [96] built a Markov chain model to analyze the tangle consensus. They classified the network into four different regimes, according to the load situations: high load (HR), low load (LR), high-to-low load (H2LR) and low-to-high-load (L2HR). In each regime, the consensus process can be divided into two stages, namely the reveal stage and accumulating stage [37]. Since the first two steady regimes HR and LR have been analyzed in [37], the authors only focus on two unsteady regimes H2LR and L2HR.

The system can be modelled as a two-dimensional stochastic process $S(t) = W(t), L(t)$ at an arbitrary time t, where $W(t)$ is the cumulative weight of a transaction observed at time t, and $L(t)$ is the total number of tips in the tangle at time t, $t = kh$, $k = 0,1,2,\ldots,\infty$. Considering that $W(t+h)$ and $L(t+h)$ are only determined by the current states $W(t)$ and $L(t)$, but not related to the earlier status, the consensus process for a new observed transaction from issuance to confirmation can be formulated as a Markov process. Furthermore, this Markov process can be formalized as a DTMC on discrete transaction arrival time intervals. Here, one step transition of the observed transaction is defined as the arrival of an incoming transaction with randomly selecting two tips for reference from $L(t)$ tips. Based on this DTMC model, the expected cumulative weight and confirmation delay at a certain time in both H2LR and L2HR can be obtained.

B. QUEUEING THEORY FOR MODELLING DLT CONSENSUSES

Queueing theory was originally proposed by Agner Krarup Erlang in 1909, to describe the Copenhagen telephone exchange. It was later developed to solve different types of system problems that involve waiting, such as customers waiting for teller service in banks. In recent years, queueing theory has been widely used to model computer networks and systems, cloud computing centers, and blockchain systems. In a blockchain system, transactions issued by clients need to wait for servers (e.g., miner, validator or orderer) to provide service (e.g., mining, validating or ordering), and finally get confirmed.

Using queueing theory, different consensus processes of DLTs can be modelled as different types of queue systems, which are named according to the Kendall's notation [113]. Within a queue system, it is possible to quantitatively answer some system performance questions such as *what is the expected number of transactions in the system*, *what is the transaction throughput of the system* and *what is the average service time (i.e., transaction time)*. In this subsection, we focus on introducing the typical queueing models (e.g., M/M/1, M/G/1 and G/M/1 queues) used for addressing these performance questions of some mainstream consensus algorithms for blockchain.

1) QUEUEING MODELS FOR PROOF-BASED CONSENSUSES

Proof-based consensus is a type of consensus mechanism that requires the network participants to provide enough proof to compete for the chance of updating the ledger. Here, we review the queue systems for modelling some popular consensus mechanisms such as PoW and PoA.

Queueing Models for PoW: In PoW-based blockchain such as Bitcoin [1], the ledger is maintained and updated by the *mining* process. In the mining process, a bunch of nodes called *miners* compete for solving very difficult puzzle-like problems, which consume a lot of computation power. Transactions issued by users are grouped into a container called a *block*, and the mining competition winner who first finds the algorithmic puzzle answer specialized for the block has the right to add the new block to the blockchain and accordingly gets incentives.

In 2017, Kawase and Kasahara [97] first built a modified M/G^B/1 queue with batch service to model the Bitcoin mining process, trying to deal with the transaction-confirmation time. In this model, transaction arrival was assumed to be a Poisson process and service time interval to be a general (or arbitrary) distribution. Arriving transactions are served in a batch manner with a maximum batch size b. In a typical M/G^B/1 queue system, an idle server starts service immediately if there are one or more customers awaiting service in the system [114]. But in this variant model, newly arriving transactions wait in the queue for getting served until the next block-generation time, even when the number of transactions is smaller than b. This is regarded as the service with multiple vacations. This is a very straightforward model description from the Bitcoin block generation and mining process based on Nakamoto's consensus, in which new transactions are grouped into a block to wait for being mined in the next block-generation time or even later on.

To analyze this queue system, the authors leveraged the joint distribution of the number of transactions in the system and the elapsed service time to derive the mean transaction-confirmation time. Then, by using the method of supplementary variables, a system of differential-difference equations was set up to formalize the problem. However, they have not successfully provided the unique solution of the differential-difference equations' system, leaving analysis of the blockchain queue system as an interesting open problem for future research.

To overcome the difficulties encountered in the original model [97], Li *et al.* introduced a new blockchain queueing model [98] by decomposing the mining process into two different exponential service stages: block-generation and blockchain-building processes. The sum of both stages' times is regarded as the *transaction-confirmation time*, also called *service time*. In this model, all Bitcoin transactions are assumed to be arriving according to a Poisson process, namely the arrival interval times follow an exponential distribution with arrival rate λ. Service times in two stages of batch services are also simply assumed to be i.i.d. and exponentially distributed with rates μ_2 and μ_1, respectively. First, each transaction enters a queue waiting room and waits for services. Then, in the first service process, a group of transactions are mined into a block with rate μ_2 and, simultaneously, a nonce is appended to the block by the mining winner. The block has a limited transaction capacity of b, also called batch size in the model. In practice, the selection of transactions may not follow a first-come-first-serve (FCFS) discipline, meaning that some latter coming transactions in the queue may be preferentially first selected into the block. But in this model, all computations are based on the FCFS discipline for the reason of simplification. Finally, a generated block with all transactions wrapped in it is attached to the blockchain in a transaction rate of μ_1. The simple blockchain queueing model is illustrated in Fig. 7.

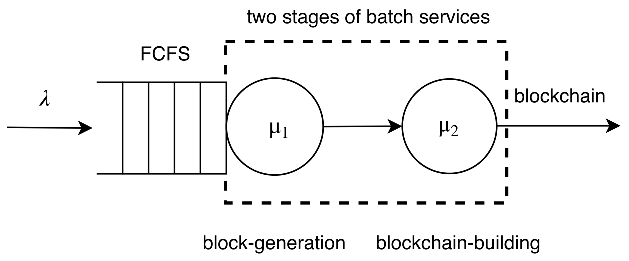


FIGURE 7. Blockchain queueing model with two batch service processes.

To analyze this queueing model, the authors defined two random variables $I(t)$ and $J(t)$ as the numbers of transactions in the block and in the queue at time t, respectively. Thus, the system can be modelled as a two-dimensional *continuous-time Markov chain (CTMC)* $X(t) = \{I(t), J(t)\}$ on the state space $\Omega = \{(i,j) : i = 0, 1, \ldots, b; j = 0, 1, 2 \ldots\}$. By analyzing the state transition diagram (see [98] for details), the only three possible transitions from an arbitrary state *(i, j)* are to state: ***(i, j+1)***, the same level; ***(0, j)***, i levels up; or ***(l, i+j−l)***, l $(1 \leq l \leq b)$ levels down. With all these characteristics, the corresponding Markov transition matrix (or infinitesimal generator) Q is a lower Hessenberg matrix, which is constructed by different repetitive small matrix blocks. Therefore, $X(t)$ is a continuous-time Markov process of GI/M/1-type. This block-structured Markov chain (the other two examples are M/M/1-type and M/G/1-type) can be solved using the matrix-analytic (or matrix-geometric) approach.

However, this model has very strong assumptions on transaction arrival and service processes. It is too specific and not suitable for many practical conditions of blockchain systems. To generalize this model, in their more recent work [99], the authors changed the transaction arrivals from Poisson to Markov arrival process (MAP), the service times from exponential to phase-type (PH), and the service discipline from FCFS to service-in-random-order. Under the new assumptions, the blockchain queueing model description keeps the same. Note that this is also a structured GI/M/1-type Markov chain. For the solution, matrix-geometric approaches are adopted to analyze and find the stable condition. This is the same as the stationary condition of the previous model. The simple expressions for the average stationary number of transactions in the queue waiting room $E(N_1)$ and the average stationary number of transactions in the block $E(N_2)$ are obtained separately.

Because of the batch service and the Service-In-Random-Order discipline for choosing transactions from the queueing waiting room into a block, the Markov chain structure becomes more complicated. This makes the computation of transaction-confirmation time very difficult. To overcome the challenge, the authors borrowed a computational technique by means of both the PH distributions and the RG factorizations [99].

There have been other blockchain queueing models proposed for analyzing the performance of PoW consensus. Ricci *et al.* [100] proposed a framework combining machine learning with queueing theory to study Bitcoin transaction delays. They introduced a simple queueing model for characterization of the transaction confirmation that can be consiered a variant of M/G/1 queue. Different from complicated mathematical derivations, the authors mainly leveraged the *operational laws* in queueing theory, such as *Little's Law* to solve the queue system. The most important result, namely average transaction delay experienced by a user, is given as $E(D) = \alpha E(B) + E(B_r)$, where α is the expected number of blocks that a user needs to wait until a transaction is confirmed, $E(B_r)$ denotes the residual time of the inter-block time, and $E(B)$ stands for the average time between block confirmations. This formalization is inspired by the standard M/G/1 queueing model, where the coefficient of the residual service time equals the system utilization. In this variant of the model, a block is always being mined, making the utilization 100% all the time.

Zhao *et al.* [101] established a type of non-exhaustive queueing model to study the *average transaction confirmation time* in a PoW-based blockchain system. For such system, any block has a size limitation, and the block cannot be confirmed during the mining process. Therefore, a non-exhaustive queue with a limited batch service and a possible zero-transaction service is naturally more suitable to capture the system features. In this queueing model, the mining process is treated as a *vacation*, and the block-verification process is regarded as a *service*. Transaction arrival is assumed to be a Poisson process with rate λ. The time duration V for a mining process and the time duration S for a block-verification process are both i.i.d. variables that follow a general distribution with distribution functions $V(t)$ and $S(t)$, respectively. *Laplace-Stieltjes transform (LST)* has been widely used in modelling both mining and block-verification processes to provide integral expressions for $E[V]$ and $E[S]$. Through a series of mathematical transformations and derivations, the authors eventually obtained the following expression for average transaction confirmation time: $E[C] = E[S] + E[V]$ (refer to [101] for details).

Queueing Models for PoA: To evaluate the performance of the mining process in Proof-of-X based blockchains, Geissler *et al.* [102] proposed a generic discrete-time $GI/GI^N/1$ queueing model. Their goal was to investigate key performance indicators, such as the mean queue size and mean transaction waiting time, and to identify significant impact factors. To make this model general, the authors abstracted the blockchain network as a single server by neglecting the information propagation delays among network nodes. Then, the model was built around a fixed-point iteration of the queue size distribution by representing the system state with queue size Q_n.

In this system model, the transaction interarrival time A follows a general distribution $a(k)$ described as $A(k) = P(A < k) = \sum_{i=0}^{k} a(k), k \in [0, \infty)$. The service time T is also assumed to follow a general distribution. Every time a new transaction arrives, the size of queue $Q(k)$ increases by one, while every block generation decreases the queue size by confirming a batch of transactions from the queue. Thus, the queue size distribution can be defined recursively, with iteration based on an embedded Markov chain with embedding times right before a block generation event. Furthermore, the distribution of key performance indicator *transaction waiting time* can be defined by the recurrence time distribution of the block generation process $r_T(x)$ and the coefficient of weighted probability $c(k)$. The corresponding expressions are obtained through recursive solutions, Little's law of queueing theory and basic probability mathematical derivations, see [102] for details. In the evaluation part, the authors obtained a good match by comparing the model data and the experimental measurements, which showed the effectiveness and accuracy of the model. Unfortunately, this general model was only validated by using a specific Ethereum implementation based on the Proof-of-Authority (PoA) consensus. It discounts the versatility of this model, since the more popular PoX consensuses such as PoW and PoS have not been examined.

2) QUEUEING MODELS FOR VOTE-BASED CONSENSUSES

Vote-based consensus is a type of high performance algorithms relying upon voting to reach agreement on transaction processing among participant nodes in a distributed system. It is the most popular consensus mechanism used in permissioned blockchain. Three widely used representatives of the vote-based consensus implementations are PBFT [115], BFT-SMaRt [116], and delegated Byzantine fault tolerance (dBFT).

Queueing Models for PBFT: The classic PBFT algorithm was firstly proposed in 1999 to solve the transmission errors and Byzantine faults in distributed systems [115]. It consists of five steps: request, pre-prepare, prepare, commit and reply. When the PBFT is adopted in constructing blockchain systems such as Hyperledger Fabric v0.6 [65], Zilliqa [117], and EOS [47], it has different implementations and/or combinations with other protocols. For example, EOS takes a hybrid consensus of combining PBFT with DPoS, to greatly reduce the required consensus time. Zilliqa uses an optimized version of classic PBFT binded with sharded PoW to achieve consensus in an efficient manner, yielding a high throughput for the blockchain system. HLF, as the most well-known permissioned enterprise-level distributed ledger platform, implements the PBFT consensus algorithm among the network peers (i.e., endorser, orderer and committer) mainly through three phases: endorsement, ordering and validation, as illustrated in Fig. 8.

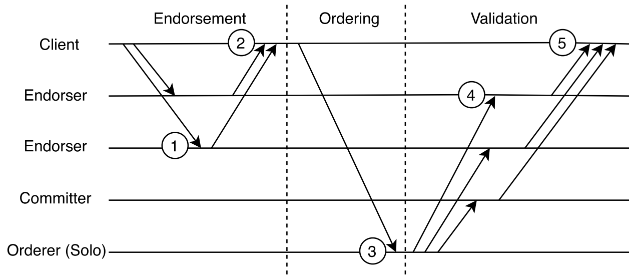


FIGURE 8. Hyperledger Fabric transaction workflow.

Phase 1. Endorsement (also called *proposal* or *execution*): ① The client generates transaction proposals and submits to endorsers for execution. ② The endorsers simulate the transactions by executing the operation previously written on the chaincode, and then return responses with signed endorsements to the client. The endorsements contain the values read or written called *read/write set* (or *rw-set*) by the chaincode.

Phase 2. Ordering: The client sends the transaction together with the endorsements to the Solo orderer for ordering service. ③ The orderer collects transactions submitted from different clients, establishes a total order on them for each channel, packages multiple transactions into blocks and generates a hash-chained sequence of blocks. As for HLF v2.0, there are three implementations of ordering peers: Solo, Kafka, and Raft.

Phase 3. Validation (also called *validation and commit*): ④ The ordered blocks are delivered to committers through gossip protocol broadcasting. All peers are committers by default, including pure committers and committers with additional endorser responsibilities. Subsequently, the peers validate each transaction contained in the received blocks. If all validations are passed, the transaction's write set is applied to the peer's world state, and the client gets a notification about the successful execution of the transaction ⑤. Otherwise, any check fail will mark the transaction as invalid, and its effects are disregarded.

Geyer *et al.* [67] modelled the Solo ordering process of HLF as an M/M^B/1 queueing system. According to the previously described three phases, transactions with endorsements arrive at the orderer at different times and are queued. While the queued transactions reach a threshold number B (called batch size), the orderer immediately provides ordering service and packages them all at once into a block. If the transactions arrive according to a Poisson process with rate λ and the ordering service time is assumed to follow exponential distribution with rate μ and FCFS discipline, the service process can be described as an M/M^{B}/1 queueing system, as shown in Fig. 9.

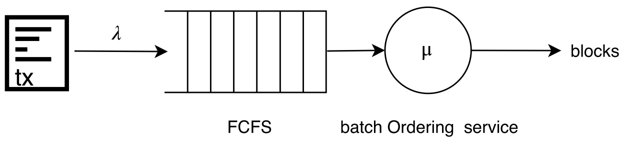


FIGURE 9. Hyperledger Fabric ordering service illustration and M/M^{B}/1 queueing model.

To solve this model, the authors borrowed the results from a well-studied general bulk service queueing model M/M$_{a,b}$/1 [118]. They simply modified the batch size range to $a = b = B$. Then, the average time spent in the ordering phase $E(T)$ can be expressed by the given parameters, among which the batch size B is approved to be significant to $E(T)$ from the numeric evaluations. This model well captures the characteristics of the ordering phase in Solo implementation. However, its shortcomings are obvious: 1) it is not suitable for Raft or Kafka implementations; and 2) it does not describe the whole transaction delay in the HLF system.

Alaslani *et al.* [103] focused on PBFT blockchain system end-to-end delay evaluation in IoT. To study the system delay, the authors built a model with two standard queues to capture the features of PBFT consensus from the system level. In this system, there are M IoT devices working as clients to send transaction requests, and K intermediate switches and R consensus replicas working together to process transactions. Since different IoT applications have different latency requirements to guarantee their service level agreement (SLA), network parameters need to be analyzed to meet the requirements. In the first part of the model, an M/G/1 queue is considered in which the maximum number of network hops K^* needs to be calculated under the application latency constrains. In the second part of the model, an M/M/1 queue is used to calculate R, the number of consensus replicas (i.e., blockchain consensus participants) needed to maintain the end-to-end requirements. Next, operational laws such as Little's law are used to analyze the network hops, and the number of consensus replicas, where three main phases (i.e., preprepare, prepare, and commit) of PBFT and its fault tolerance capability f out of $N = 3f + 1$ replicas are taken into consideration.

Fork-Join Queue for Vote-Based Consensus: In vote-based permissioned blockchain systems, transactions are broadcast to all authenticated voting peers of the P2P overlay after being proposed. These voting peers, called miner nodes or validators, are selected and authorized to validate transactions, generate new blocks and record data to the local ledger if a transaction gets enough validation votes, e.g., k out of n. For example, in the PBFT, a block is accepted and recorded if $2f + 1$ out of $n = 3f + 1$ peers independently agree on the block of transactions, where f is the maximum number of Byzantine fault peers this system can tolerate.

The idea of leveraging an (n, k) fork-join queue to model vote-based blockchain is based on the fact that the service process of this queue system matches well with the above-mentioned transaction propagation and validation procedure. In an (n, k) fork–join queue, the incoming jobs are split/forked on arrival for simultaneous and independent service by numerous servers and joined before departure. While in a vote-based blockchain system, if we consider the confirmation of a transaction as a big job requiring enough validations from n nodes, this job can be split into n sub-tasks, associated with being broadcast to n nodes and being validated independently at the same time. Once any k out of n sub-tasks are finished, they are joined to finish the service and make the transaction confirmed and recorded to the local ledger. The remaining $n-k$ sub-tasks keep executing until being finished. This is called a non-purging (n, k) fork-queue. By contrast, a purging (n, k) fork-join queue removes all remaining sub-tasks of a job from both sub-queues and service nodes once it receives the job's kth answer.

In the literature, this model is highly prevalent for performance modelling (e.g., estimating the sojourn time of jobs in the queues) of parallel and distributed systems. Recently, it has been found effective for use in studying the delay performance of the synchronization process of the vote-based permissioned blockchain systems [104]. A typical non-purging (n, k) fork-join queueing model is illustrated in Fig. 10.

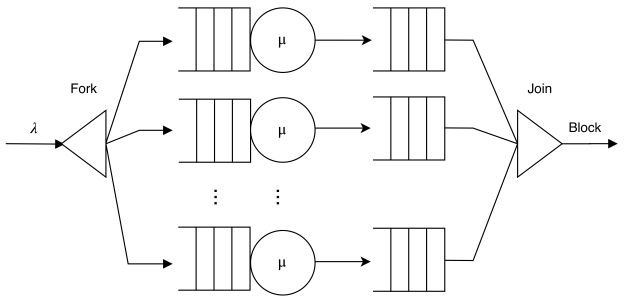


FIGURE 10. A typical fork-join queueing model. All blockchain voting nodes are homogeneous with the common service rate μ.

Even though few analytical results exist for fork–join queues, various approximation solutions are known. An example is the linear transformation approach [119], which can be used to approximately compute the sojourn time $t(n, k)$ of a general non-purging (n, k) fork-join queue for the vote-based blockchain system.

3) FLUID QUEUE FOR IOTA TANGLE

In queueing theory, a fluid queue (also called fluid model) is a mathematical model used to describe the fluid level in a reservoir, for which the periods of filling and emptying are randomly determined. It can be viewed as a large tank connected to a series of pipes that pour fluid into the tank and a series of pumps that remove fluid from the tank. The capacity of this tank is typically assumed to be infinite. The fluid level $X(t)$ of this tank at time t is a random variable that can be calculated if the fluid arrival and leaving rates are given.

This model was successfully used to describe the dynamic behaviour of the IOTA tangle [105]. First, a fluid model was heuristically built based on some requisite stochastic models and the assumptions on the transaction arrival rate. Through solving the proposed delay differential equations system, the authors analyzed the stability of conflicts, which impacted the performance in return.

C. STOCHASTIC PETRI NETS FOR MODELLING DLT CONSENSUSES

Another type of commonly used analytical tool for BFT-based consensus performance modelling is *stochastic Petri net (SPN)*, especially its variants *generalized stochastic Petri net (GSPN)* and *stochastic reward net (SRN)*. Petri nets (PNs) are a type of powerful mathematical modelling language used to model and simulate discrete-event distributed systems. They are graphs consisting of two types of nodes: places and transitions, which represent variables of system states represented by circles and actions made by the system represented by rectangles. When the *firing* times of all transitions are exponentially distributed (timed transitions), the model is called SPN. Built on SPN, a GSPN allows transitions to have zero firing times (immediate transitions) and *inhibitor arc* – an arc from a place to a transition that inhibits the firing of the transition when the input place is not empty. According to the literature, any GSPN model can be converted to an equivalent CTMC, and vice versa. At the net level, an SRN substantially improves the modelling power of the GSPN by adding guard functions, marking dependent arc multiplicities, general transition priorities, and reward rates.

HLF V1.0+ adopts a highly modular architecture design by decomposing the transaction process into three main stages as shown in Fig. 8. They can be also refined into five phases, namely HTTP, endorsement, ordering, validation & committing, and response [106]. HLF's modular design makes it possible to separately build a model for each phase and then cascade them to analyze the performance from the net/system level. There have been two studies on HLF performance analysis using GSPN [106] and SRN [107], respectively. Both follow these general steps: 1) clarify transaction process steps and the business logic behind them; 2) create the associated transition diagrams of Petri nets according to the corresponding rules under reasonable assumptions; 3) translate to Markov chains for analytical solutions or directly leverage mathematical tools for numerical simulation solutions. The second step is critical because it bridges the real system to an analytical model and paves the way to solutions for the performance indicators such as transaction throughput, latency, average queue length and utilization. Here, we focus on the Petri nets' transition diagrams of the ordering phase from the two reviewed studies, as shown in Fig. 11.

In Fig. 11 (a), the ordering service starts with taking endorsed transactions as inputs under the assumptions of the

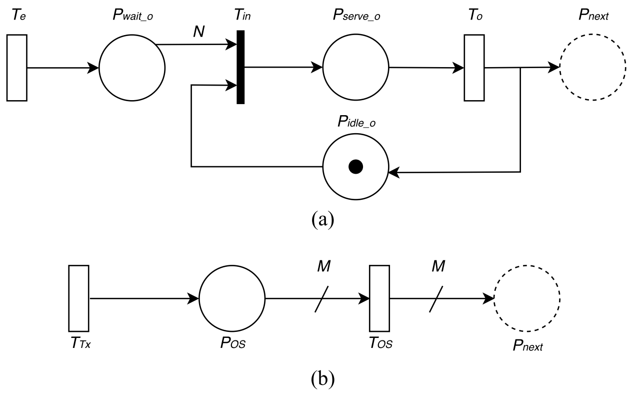


FIGURE 11. SPN models for ordering service in HLF V1.0+: (a) GPSN [106] (b) SRN [107].

exponentially distributed request arrival and constant size of each transaction. The symbols in the figure are interpreted as follows: T_e is a transition signifying the arrival of an endorsed transaction. P_{wait_o} is a place signifying the transaction is queuing, the number of token $\#(P_{wait\_o})$ denotes the queuing length. N is the batch processing size in number of transactions. P_{serve_o} is a place signifying transactions are being ordered. P_{idle_o} is a place signifying the server is idle now, the number of token $\#(P_{idle\_o})$ denotes the number of idle servers. T_{in} is an immediate transition whose enable predicate is $\#(P_{wait\_o}) > 0$ & $\#(P_{idle\_o}) > 0$, which means there are idle servers and queuing transactions. P_{next} is a place signifying the next processing phase.

Similarly, other phases can be modelled by following the same methodology. Consequently, the proposed analytic system model based on GSPN indicates that the HLF system is composed of multiple successive M/M/1 queue networks, and the system throughput is equal to the lowest throughput of all those phases. Using a tool embedded in Matlab named *pntool*, this system can be numerically solved to determine the latency and throughput.

The second part of Fig. 11 describes a simple SRN model for HLF ordering service in a network with one client, two endorsers and one peer running the validation logic. After the client receives a response from both endorsing peers, it sends the endorsed transaction to the ordering service (transition T_{Tx}), specified by a token deposited in place P_{OS}. When the number of pending transactions reaches block size (denoted by M) or block timeout for general, a number of transactions are ordered into a block (transition T_{OS}). The block is delivered to the committing peers (place T_{next}) for validations, such as VSCC validation and MVCC validation. Finally, all successfully validated transactions in the block are recorded into the local ledger. As for solving this SRN model, one can use the simulation approach called Stochastic Petri net Package (SPNP) [120] to numerically find answers for the following performance metrics.

- *throughput:* corresponds to the rate of each transition, using function *rate()* in SPNP to capture. E.g., the rate of transition T_{Ledger} signifies the block throughput of the system, which can be used to multiply by M to obtain transaction throughput.

- *utilization:* computed by the probability that the corresponding transition in SRN is enabled, using function *enabled()*, or computed using reward functions for transitions with function-dependent marking rate (such as TVSCC).
- *mean queue length:* obtained by the number of tokens in the corresponding phase, using function *mark()*. E.g., the mean number of tokens in place P_{OS} indicates the mean queue length at the ordering service.

D. OTHER MODELS IN DLT PERFORMANCE MODELLING

Besides the stochastic models described earlier, there have been other analytical models proposed for analyzing blockchain performance. For example, a prediction model [108] derived from the core Ethereum's structure called *World State* was proposed to provide companies with a more accurate estimation of performance and required storage. By analyzing the modified Merkle Patricia tree (MPT), which is the implementation of the *World State* in Ethereum, the expectation and the max tree height were derived as a function of the total number of transactions n. These results linked to the performance and storage, which were meaningful for decision making and early warnings. Another study [109] adopted *stochastic network models* to analyze the overall *block generation rate* for the PoW-based Ethereum. Through this model, the blockchain evolution and dynamics can be captured and used to analyze the impact of the block dissemination delay and hashing power of the member nodes on the *block generation rate*.

Random graph (also called Erdős-Rényi model) is a powerful mathematical tool first introduced by Erdos [121] and Bollobás and Béla [122] to model and analyze complex networks. It has properties suitable for modelling the peer-to-peer overlay networks used by blockchain systems [110]. There are two main variants of the Erdős-Rényi model. One of them is $G_p(N)$, which is a graph constructed by randomly connecting nodes. Each edge is included in the graph with probability p independent from every other edge. Shahsavari *et al.* [110] presented a random graph using $G_p(N)$ to model the Bitcoin blockchain network, where N is the total number of nodes, and p refers to the independent probability that there exists a link between any two observed nodes in the peer-to-peer overlay network. Based on the well-established random graph analysis results, some key performance measures can be derived in terms of *block dissemination delay* and *traffic overhead*.

V. FINDINGS AND SUGGESTIONS FOR FUTURE RESEARCH

In this section, we summarize the main findings from previous evaluation sections. First, we discuss the findings from the empirical and analytical evaluations. We then take a look at the performance bottlenecks identified from all reviewed solutions. Finally, we point out some open issues and provide suggestions for future research.

A. FINDINGS FOR EMPIRICAL ANALYSIS

1) PERFORMANCE METRICS AND WORKLOADS

The evaluated performance metrics can be divided into two categories: macro (or overall) metrics and micro (or detailed) metrics. *Macro metrics* provide an overview of the system's performance for users from the application level, such as transaction throughput, latency, scalability, fault tolerance, transactions per CPU/memory second/disk IO/network data. The first two metrics (transaction throughput and latency) are evaluated most frequently, over all blockchains. *Micro metrics* depict the performance of different subprocesses of transactions or specific layers in the blockchain abstract model for developers, such as peer discovery rate, RPC response rate, transaction propagating rate, contract execution time, state updating time, consensus-cost time, encryption and hash function efficiency. Both macro and micro metrics are evaluated under well-designed workloads.

In blockchain performance benchmarking or monitoring frameworks, these workloads have been designed to evaluate the performance of different layers of blockchain. *Macro workloads*, such as YCSB, Smallbank, EtherId, Doubler and WavesPresale, are designed to evaluate the application layer in blockchain. *Micro workloads*, such as DoNothing, Analytics, IOHeavy and CPUHeavy, are designed to evaluate lower layers of blockchain, including execution, data and consensus layers [49].

In general, there are two popular ways to generate workloads for experiment-based performance evaluation. One is to construct a synthetic application with commonly used functions (e.g., CreateAccount, IssueMoney and TransferMoney), and leverage a client node to send requests of transactions (i.e., implemented functions) to a blockchain system [78]. The other is to leverage HTTP performance testing toolkit for generating requests, for example, using the *loadtest* library of Node.js to specify an HTTP request as a JSON-formated object, and constructing workloads for blockchains as separate JSON objects [77], [84].

2) EVALUATED BLOCKCHAINS

HLF (v0.6 with PBFT and v1.0+ with BFT-SMaRt), private Ethereum (Geth with PoW and Parity with PoA/PoW) and Ripple with XRP consensus are the most often comparatively evaluated blockchain platforms [49], [72], [78], [80]. Among them, HLF and Ripple can reach 1,000+ TPS within a small network and outperform the Ethereum platforms in terms of throughput and latency, under both macro and micro benchmark workloads. However, because of the underlying consensus algorithms they use, both HLF and Ripple fail to scale beyond a certain number of nodes in the network (e.g., 16 [49] for HLF v0.6). For HLF, it is well-known that BFT-based consensuses (e.g., PBFT and BFT-SMaRt) rely on a leader for processing transactions, which may act as a bottleneck and cause performance limitations. For Ripple, a limited and fixed number of validators receive and process numerous transaction requests, and finally fail to scale when the number of requests goes beyond the capability of the validators. This conclusion is shared by a number of early evaluation studies such as [49], [77], [78], [80]. Between the different versions of HLF, its new release v1.0+ has better performance than v0.6 [62] across all evaluated macro metrics such as execution time, latency, throughput and scalability. In addition, another blockchain proposed for IoT (i.e., Tendermint) outperforms HLF V0.6 and Ripple on both the throughput and the latency [84].

It is worth noting that we did not encounter any improvement solutions such as off-chain, side-chain, concurrent execution and sharding in the evaluated blockchain systems. In fact, many of the proposed solutions only exist at the conceptual stage at the time of writing this survey. Some of them provide a brief comparative evaluation and analysis under a specific use case for the purpose of proof-of-concept, but lack a systematic evaluation in a meaningful manner to demonstrate their effectiveness and efficiency.

3) CONSENSUS FINALITY

Consensus finality refers to the deterministic property of a blockchain where a block is considered confirmed once it is appended to the ledger. BFT-based blockchains are all with consensus finality, while those PoW-based are usually not. This property has a direct impact on the transaction latency. For example, Bitcoin usually requires six successive confirmations as a secure finality that a transaction will not end up being pruned and removed from the blockchain, which makes the latency reach an unacceptable time of almost one hour. In contrast, HLF with BFT-based consensuses can finalize a transaction within seconds right after it is appended to the ledger. Therefore, BFT-based blockchains have an obvious advantage over PoW-based blockchain systems in terms of performance.

B. FINDINGS FOR ANALYTICAL MODELLING

Performance Modelling Strategies: Most models neglect information propagation delays in the network and simply collapse the whole network into a single node that provides service to process and confirm transactions. These models are usually queue systems that provide bulk services such as $M/M^B/1$ and $M/G^B/1$ queues. Only a small portion of models consider the system as separate disjoint nodes and take the network latency among network nodes into consideration. They aim to calculate system end-to-end output (e.g., delays) using queue networks or by cascading different queues such as M/G/1 and M/M/1 together to model the blockchain network.

An (n, k) fork-join queue combines both modelling strategies. It first regards the system as a single server when the system receives a job request. Then, it splits the job into several sub-tasks for independent and simultaneous processes on different network nodes. In the joint phase, process results are collected from different nodes to finish the original job (e.g., block validation).

C. IDENTIFIED PERFORMANCE BOTTLENECKS

From the perspective of users or managers, performance evaluation results can be used for decision making on blockchain system selection. Developers and system designers, on the other hand, may care more about the identified bottlenecks rather than the comparison results. They can analyze these bottlenecks and propose solutions for further performance optimization. All bottlenecks identified in the reviewed papers are listed in Table 7.

As we can see, most bottlenecks are still unresolved. This means that corresponding effective solutions to solve the performance problems have not yet been found. Another observation is that most bottlenecks are identified by empirical analysis, which can be attributed to two reasons. First, there are more empirical analyses conducted than performance modelling. Second, due to the involved mathematical expressions, analytical modelling is much more difficult than experimental solutions in exploring the impact of design parameters. In blockchain performance modelling, even one simple extra parameter can significantly increase the model complexity. Therefore, empirical analysis becomes more efficient and popular in bottleneck identification than its modelling counterpart.

D. OPEN ISSUES AND FUTURE DIRECTIONS

As a fundamental component of blockchain research, performance evaluation plays an important role in boosting blockchain applications. Although numerous blockchain improvements have been proposed and implemented, only a small number of them have been well evaluated. The evaluation methods also need more analysis and explorations. Here, we identify some open issues and suggest potential directions for future research in this area.

- For empirical analysis, difficulties lie in comparative evaluation among different blockchain platforms, especially for those with very different consensus algorithms and data structures. The main reason is the lack of interface standards in running workloads. For example, when evaluating blockchain platforms for IoT such as HLF 1.0, Ripple and IOTA, it is difficult to design a common interface for uploading workload. Since smart contracts are not supported by Ripple or IOTA, one solution is to design an equivalent workload such as transferring a unity amount from account A to another account B [77]. However, this approach has limited extensibility, and requires to deploy a dedicated workload for each blockchain under evaluation. Thereby, there is a great potential for future research to develop more extensible tools for comparative evaluation of blockchain platforms.
- Many methods of experimental analysis rely on RPCs to communicate with blockchain consensus nodes and collect transaction statistic data (e.g., the total number

TABLE 7. Identified performance bottlenecks for different blockchain systems.

Blockchain	Bottlenecks Identified	Evaluation Approaches	Latest State
Ethereum v1.5.9	peer discovery, transactions propagation, consensus-cost	Monitoring [61]	Unresolved
Geth v1.4.18	consensus protocols	Benchmarking [49]	Unresolved
HLF v0.6.0	consensus protocols	Benchmarking [49]	Unresolved
Parity v1.6.0	transaction signing	Benchmarking [49]	Unresolved
HLF v1.0	endorsement policy verification, sequential policy validation of transactions in a block, and state validation and commit (with CouchDB)	Experimental analysis [64]	Resolved (HLF v1.1)
Byteball	data storage which is a relational database	Benchmarking [56]	Unresolved
HLF v1.0	no parallel transaction processing on the committing peer	Experimental analysis [56]	Unresolved
HLF	ordering service	Experimental analysis [67]	Unresolved
Private Ethereum	module responsible for reading and writing data	Experimental analysis [79]	Unresolved
Private Ethereum	consensus mechanism	Experimental analysis [80]	Unresolved
HLF	consensus mechanism	Experimental analysis [80]	Unresolved
HLF v1.0+	transmission from client to the ordering service and ledger write	Analytical modelling [107]	Resolved
HLF v1.2	committing phase if the number of transactions in a block is small and endorsement phase if it is large	Analytical modelling [106]	Unresolved

of confirmed transactions of certain duration). Although the RPC API protocols (e.g., gRPC and JSON-RPC) claim to be efficient, they still induce extra overhead onto the consensus peers [61], which is counted as the peer consumption and in turn makes the evaluation results inaccurate. Therefore, a more light-weight and low-overhead data collection approach, such as log-based approach [61], deserves more attention in the future.

- RPC methods are widely used for data collection in empirical performance evaluation of blockchain systems. For micro metrics and micro workload design, it is challenging to decouple the impact from other layers. For example, two queries on transaction values are designed to evaluate the *data model* performance. For Ethereum, both queries can be easily implemented via invoking JSON-RPC APIs. However, for HLF, a chaincode (VersionKVStore) must be implemented as there are no APIs to query historical states in the system. Inevitably, this involves execution of a smart contract making the evaluation inaccurate by adding extra overhead. Therefore, for detailed evaluation of performance metrics in specific blockchain abstraction layers, it is an open issue to design a reasonable workload that alleviates the impact of other layers and improves accuracy.
- Besides the classic blockchain systems such as HLF and Ethereum, there is an urgent need for evaluating the performance of their proposed improvements. For example, *sharding* claims to be a promising solution and has been implemented in many blockchains. However, there is no evaluation work for comparing different shard-based blockchain systems. Different solutions, such as sharding v.s. DAG and off-chain v.s. side-chain also need to be comparatively evaluated. In addition, it would be beneficial to combine empirical and analytical approaches in blockchain performance evaluation in the future.

VI. CONCLUSION

As blockchain has matured to receive more and more attention, its performance problems (e.g., low throughput and high latency) have became critical. To resolve these issues, there have been many improvements proposed, from system level optimization to new efficient consensus protocols. However, such blockchain modifications need to be evaluated in a meaningful manner to demonstrate their performance advantages. In this paper, we present a systematic survey covering existing blockchain performance evaluation approaches. From the high level perspective, they can be categorized into *empirical* and *analytical* evaluation methods.

The empirical analysis can be further divided into four groups: performance benchmarking, monitoring, experimental analysis and simulation. Three popular benchmark frameworks (i.e., Blockbench, DAGbench and Hyperledger Caliper) are introduced and comparatively analyzed. Performance monitoring is recognized as the best solution for performance evaluation of public blockchain.

Analytical modelling approaches are more powerful than empirical solutions especially for analyzing the consensus layer of blockchain system. There are three main types of modelling approaches compared in this survey: Markov chains, queueing models and stochastic Petri nets. This comparison can provide directions for selecting blockchain evaluation approach suitable for given purpose.

We also summarized the results of surveyed performance evaluation studies and identified the bottlenecks of major blockchain platforms. The survey concludes with identification of open issues and ascertainment of future research directions in this important area.

ACKNOWLEDGMENT

The authors would also like to thank Compute Canada, SAVI, and Cybera for the support of this research through their cloud services.

REFERENCES

[1] S. Nakamoto and A. Bitcoin. (2008). *A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[2] Q. K. Nguyen, "Blockchain—A financial technology for future sustainable development," in *Proc. 3rd Int. Conf. Green Technol. Sustain. Develop. (GTSD)*, Nov. 2016, pp. 51–54.
[3] L. Cocco, A. Pinna, and M. Marchesi, "Banking on blockchain: Costs savings thanks to the blockchain technology," *Future Internet*, vol. 9, no. 3, p. 25, Jun. 2017.
[4] M. Hölbl, M. Kompara, A. Kamišalić, and L. N. Zlatolas, "A systematic review of the use of blockchain in healthcare," *Symmetry*, vol. 10, no. 10, p. 470, Oct. 2018.
[5] C. C. Agbo, Q. H. Mahmoud, and J. M. Eklund, "Blockchain technology in healthcare: A systematic review," *Healthcare*, vol. 7, no. 2, p. 56, Apr. 2019.
[6] L. A. Linn and M. B. Koo, "Blockchain for health data and its potential use in health it and health care related research," in *Proc. ONC/NIST Use Blockchain Healthcare Res. Workshop*. Gaithersburg, MD, USA: ONC/NIST, 2016, pp. 1–10.
[7] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *Proc. IEEE 18th Int. Conf. e-Health Netw., Appl. Services (Healthcom)*, Sep. 2016, pp. 1–3.
[8] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3690–3700, Aug. 2018.
[9] E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, and C. Weinhardt, "A blockchain-based smart grid: Towards sustainable local energy markets," *Comput. Sci.-Res. Develop.*, vol. 33, nos. 1–2, pp. 207–214, Feb. 2018.
[10] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 5, pp. 840–852, Sep. 2018.
[11] G. Perboli, S. Musso, and M. Rosano, "Blockchain in logistics and supply chain: A lean approach for designing real-world use cases," *IEEE Access*, vol. 6, pp. 62018–62028, 2018.
[12] E. Tijan, S. Aksentijević, K. Ivanić, and M. Jardas, "Blockchain technology implementation in logistics," *Sustainability*, vol. 11, no. 4, p. 1185, Feb. 2019.
[13] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2016, pp. 106–125.
[14] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.
[15] J. Reed, *Litecoin: An Introduction to Litecoin Cryptocurrency and Litecoin Mining*. Scotts Valley, CA, USA: CreateSpace Independent Publishing Platform, 2017.
[16] J. Teutsch and C. Reitwießner, "Truebit: A scalable verification solution for blockchains," White Papers, 2018.
[17] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *Proc. 27th USENIX Secur. Symp. (USENIX Security)*, 2018, pp. 1353–1370.
[18] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," White Papers, 2016.
[19] Raiden Network. (2018). *Fast, Cheap, Scalable Token Transfers for Ethereum*. Accessed: Jul. 7, 2020. [Online]. Available: https://raiden.network
[20] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille. (2014). *Enabling Blockchain Innovations With Pegged Sidechains*. [Online]. Available: http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains
[21] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," White Paper, 2017, pp. 1–47. Accessed: Jul. 7, 2020. [Online]. Available: https://plasma.io/plasma-deprecated.pdf
[22] P. Gazi, A. Kiayias, and D. Zindros, "Proof-of-stake sidechains," IACR Cryptol. ePrint Arch., White Papers, 2018, p. 1239.
[23] A. Kiayias and D. Zindros, "Proof-of-work sidechains," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2019, pp. 21–34.
[24] T. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, "Adding concurrency to smart contracts," *Distrib. Comput.*, vol. 33, pp. 209–225, 2020.
[25] L. Yu, W.-T. Tsai, G. Li, Y. Yao, C. Hu, and E. Deng, "Smart-contract execution with concurrent block building," in *Proc. IEEE Symp. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2017, pp. 160–167.
[26] P. S. Anjana, S. Kumari, S. Peri, S. Rathor, and A. Somani, "An efficient framework for optimistic concurrent execution of smart contracts," in *Proc. 27th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, Feb. 2019, pp. 83–92.
[27] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 17–30.
[28] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 583–598.
[29] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Jan. 2018, pp. 931–948.
[30] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. 16th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2019, pp. 95–112.
[31] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, 2019, pp. 123–140.
[32] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2015, pp. 528–547.
[33] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A fast and scalable cryptocurrency protocol," IACR Cryptol. ePrint Arch., White Papers, 2016, p. 1159.
[34] Y. Sompolinsky and A. Zohar, "PHANTOM: A scalable blockdag protocol," IACR Cryptol. ePrint Arch., White Papers, 2018, p. 104.
[35] C. Li, P. Li, D. Zhou, W. Xu, F. Long, and A. Yao, "Scaling Nakamoto consensus to thousands of transactions per second," 2018, *arXiv:1805.03870*. [Online]. Available: http://arxiv.org/abs/1805.03870
[36] S. D. Lerner, "DagCoin: A cryptocurrency without blocks," White Paper, 2015. Accessed: Jul. 7, 2020. [Online]. Available: https://dagcoin.org/wpcontent/uploads/2019/07/Dagcoin_White_Paper.pdf
[37] S. Popov, "The tangle," White Papers, 2016, p. 131.
[38] A. Churyumov. (2016). *Byteball: A Decentralized System for Storage and Transfer of Value*. [Online]. Available: https://byteball.org/Byteball.pdf
[39] C. LeMahieu. (2018). *Nano: A Feeless Distributed Cryptocurrency Network*. Accessed: Mar. 24, 2018. [Online]. Available: https://nano.org/en/whitepaper
[40] S. Kim, Y. Kwon, and S. Cho, "A survey of scalability solutions on blockchain," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2018, pp. 1204–1207.
[41] S. Rouhani and R. Deters, "Security, performance, and applications of smart contracts: A systematic survey," *IEEE Access*, vol. 7, pp. 50759–50779, 2019.
[42] X. Zheng, Y. Zhu, and X. Si, "A survey on challenges and progresses in blockchain technologies: A performance and security perspective," *Appl. Sci.*, vol. 9, no. 22, p. 4731, 2019.
[43] R. Wang, K. Ye, and C.-Z. Xu, "Performance benchmarking and optimization for blockchain systems: A survey," in *Proc. Int. Conf. Blockchain*. Cham, Switzerland: Springer, 2019, pp. 171–185.
[44] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020.
[45] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, "Survey: Sharding in blockchains," *IEEE Access*, vol. 8, pp. 14155–14181, 2020.
[46] V. Acharya, A. E. Yerrapati, and N. Prakash, *Oracle Blockchain Quick Start Guide: A Practical Approach to Implementing Blockchain in Your Enterprise*. Birmingham, U.K.: Packt, 2019.
[47] EOS. IO. (2017). *EOSIO Technical White Paper*. Accessed: Dec. 18, 2017. [Online]. Available: https://github.com/EOSIO/Documentation
[48] CB Insights. (2019). *Banking is Only the Beginning: 42 Big Industries Blockchain Could Transform*. Accessed: Feb. 24, 2020 [Online]. Available: https://www.cbinsights.com/research/industries-disrupted-blockchain//
[49] T. 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 (SIGMOD)*, 2017, pp. 1085–1100.
[50] S. T. Leutenegger and D. Dias, "A modeling study of the TPC-C benchmark," *ACM SIGMOD Rec.*, vol. 22, no. 2, pp. 22–31, Jun. 1993.
[51] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proc. 1st ACM Symp. Cloud Comput. (SoCC)*, 2010, pp. 143–154.
[52] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux, "OLTP-bench: An extensible testbed for benchmarking relational databases," *Proc. VLDB Endowment*, vol. 7, no. 4, pp. 277–288, Dec. 2013.

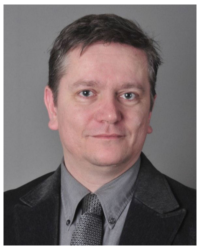
[53] V. Abramova and J. Bernardino, "NoSQL databases: MongoDB vs cassandra," in *Proc. Int. C* Conf. Comput. Sci. Softw. Eng. (CSE)*, 2013, pp. 14–22.
[54] Y. Abubakar, T. S. Adeyi, and I. G. Auta, "Performance evaluation of NoSQL systems using YCSB in a resource austere environment," *Int. J. Appl. Inf. Syst.*, vol. 7, no. 8, pp. 23–27, Sep. 2014.
[55] H. Matallah, G. Belalem, and K. Bouamrane, "Experimental comparative study of NoSQL databases: HBASE versus MongoDB by YCSB," *Int. J. Comput. Syst. Sci. Eng.*, vol. 32, no. 4, pp. 307–317, 2017.
[56] Z. Dong, E. Zheng, Y. Choon, and A. Y. Zomaya, "DAGBENCH: A performance evaluation framework for DAG distributed ledgers," in *Proc. IEEE 12th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2019, pp. 264–271.
[57] "Hyperledger blockchain performance metrics white paper V1.01," HyperLedger Found., Hyperledger Perform. Scale Working Group, White Paper, 2018. Accessed: Jul. 7, 2020. [Online]. Available: https://www.hyperledger.org/wpcontent/uploads/2018/10/HL_Whitepaper_Metrics_PDFVersion.pdf
[58] A. Aldweesh, M. Alharby, M. Mehrnezhad, and A. Van Moorsel, "OpBench: A CPU performance benchmark for ethereum smart contract operation code," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 274–281.
[59] A. Aldweesh, M. Alharby, E. Solaiman, and A. van Moorsel, "Performance benchmarking of smart contracts to assess miner incentives in ethereum," in *Proc. 14th Eur. Dependable Comput. Conf. (EDCC)*, Sep. 2018, pp. 144–149.
[60] M. T. Oliveira, G. R. Carrara, N. C. Fernandes, C. V. N. Albuquerque, R. C. Carrano, D. S. V. Medeiros, and D. M. F. Mattos, "Towards a performance evaluation of private blockchain frameworks using a realistic workload," in *Proc. 22nd Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, Feb. 2019, pp. 180–187.
[61] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A detailed and real-time performance monitoring framework for blockchain systems," in *Proc. 40th Int. Conf. Softw. Eng. Softw. Eng. Pract. (ICSE-SEIP)*, 2018, pp. 134–143.
[62] Q. Nasir, I. A. Qasse, M. A. Talib, and A. B. Nassif, "Performance analysis of hyperledger fabric platforms," *Secur. Commun. Netw.*, vol. 2018, pp. 1–14, Sep. 2018.
[63] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat, and S. Chatterjee, "Performance characterization of hyperledger fabric," in *Proc. Crypto Valley Conf. Blockchain Technol. (CVCBT)*, Jun. 2018, pp. 65–74.
[64] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *Proc. IEEE 26th Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst. (MASCOTS)*, Sep. 2018, pp. 264–276.
[65] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, pp. 1–15.
[66] T. S. L. Nguyen, G. Jourjon, M. Potop-Butucaru, and K. L. Thai, "Impact of network delays on hyperledger fabric," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 222–227.
[67] 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 *Proc. Int. Conf. Netw. Syst. (NetSys)*, Mar. 2019, pp. 1–8.
[68] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 536–540.
[69] S. Wang, "Performance evaluation of hyperledger fabric with malicious behavior," in *Proc. Int. Conf. Blockchain*. Cham, Switzerland: Springer, 2019, pp. 211–219.
[70] 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 *Proc. 18th Int. Symp. Parallel Distrib. Comput. (ISPDC)*, Jun. 2019, pp. 50–57.
[71] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1366–1385, Jul. 2018.
[72] S. Rouhani and R. Deters, "Performance analysis of ethereum transactions in private blockchain," in *Proc. 8th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Nov. 2017, pp. 70–74.
[73] R. Yasaweerasinghelage, M. Staples, and I. Weber, "Predicting latency of blockchain-based systems using architectural modelling and simulation," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Apr. 2017, pp. 253–256.
[74] C. Rathfelder and B. Klatt, "Palladio workbench: A quality-prediction tool for component-based architectures," in *Proc. 9th Work. IEEE/IFIP Conf. Softw. Archit.*, Jun. 2011, pp. 347–350.
[75] M. Bez, G. Fornari, and T. Vardanega, "The scalability challenge of ethereum: An initial quantitative analysis," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2019, pp. 167–176.
[76] C. Fan, H. Khazaei, Y. Chen, and P. Musilek, "Towards a scalable DAG-based distributed ledger for smart communities," in *Proc. IEEE 5th World Forum Internet Things (WF-IoT)*, Apr. 2019, pp. 177–182.
[77] R. Han, V. Gramoli, and X. Xu, "Evaluating blockchains for IoT," in *Proc. 9th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Feb. 2018, pp. 1–5.
[78] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2017, pp. 1–6.
[79] S. Chen, J. Zhang, R. Shi, J. Yan, and Q. Ke, "A comparative testing on performance of blockchain and relational database: Foundation for applying smart technology into current business systems," in *Proc. Int. Conf. Distrib., Ambient, Pervas. Interact.* Cham, Switzerland: Springer, 2018, pp. 21–34.
[80] Y. Hao, Y. Li, X. Dong, L. Fang, and P. Chen, "Performance analysis of consensus algorithm in private blockchain," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 280–285.
[81] H. M. A. Aljassas and S. Sasi, "Performance evaluation of proof-of-work and collatz conjecture consensus algorithms," in *Proc. 2nd Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, May 2019, pp. 1–6.
[82] R. Deloin, "Proof of collatz conjecture," *Asian Res. J. Math.*, vol. 14, no. 2, pp. 1–18, Jun. 2019.
[83] S. Benahmed, I. Pidikseev, R. Hussain, J. Lee, S. M. A. Kazmi, A. Oracevic, and F. Hussain, "A comparative analysis of distributed ledger technologies for smart contract development," in *Proc. IEEE 30th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2019, pp. 1–6.
[84] R. Han, G. Shapiro, V. Gramoli, and X. Xu, "On the performance of distributed ledgers for Internet of Things," *Internet Things*, vol. 10, Jun. 2020, Art. no. 100087.
[85] S. Park, S. Oh, and H. Kim, "Performance analysis of DAG-based cryptocurrency," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2019, pp. 1–6.
[86] S. Chandel, W. Cao, Z. Sun, J. Yang, B. Zhang, and T.-Y. Ni, "A multi-dimensional adversary analysis of RSA and ECC in blockchain encryption," in *Proc. Future Inf. Commun. Conf.* Cham, Switzerland: Springer, 2019, pp. 988–1003.
[87] J. Ferreira, M. Antunes, M. Zhygulskyy, and L. Frazão, "Performance of hash functions in blockchain applied to IoT devices," in *Proc. 14th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2019, pp. 1–7.
[88] T. M. Fernández-Caramés and P. Fraga-Lamas, "A review on the use of blockchain for the Internet of Things," *IEEE Access*, vol. 6, pp. 32979–33001, 2018.
[89] M. Alharby and A. van Moorsel, "Blocksim: A simulation framework for blockchain systems," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 135–138, 2019.
[90] S. Pandey, G. Ojha, B. Shrestha, and R. Kumar, "BlockSIM: A practical simulation tool for optimal network design, stability and planning," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2019, pp. 133–137.
[91] C. Faria and M. Correia, "BlockSim: Blockchain simulator," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 439–446.
[92] M. Zander, T. Waite, and D. Harz, "DAGsim: Simulation of DAG-based distributed ledger protocols," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 118–121, Jan. 2019.
[93] M. Bottone, F. Raimondi, and G. Primiero, "Multi-agent based simulations of block-free distributed ledgers," in *Proc. 32nd Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA)*, May 2018, pp. 585–590.
[94] B. Kusmierz, W. Sanders, A. Penzkofer, A. Capossele, and A. Gal, "Properties of the tangle for uniform random and random walk tip selection," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 228–236.
[95] D. Huang, X. Ma, and S. Zhang, "Performance analysis of the raft consensus algorithm for private blockchains," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 1, pp. 172–181, Jan. 2020.
[96] B. Cao, S. Huang, D. Feng, L. Zhang, S. Zhang, and M. Peng, "Impact of network load on direct acyclic graph based blockchain for Internet of Things," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery (CyberC)*, Oct. 2019, pp. 215–218.

[97] Y. Kawase and S. Kasahara, "Transaction-confirmation time for bitcoin: A queueing analytical approach to blockchain mechanism," in *Proc. Int. Conf. Queueing Theory Netw. Appl.* Cham, Switzerland: Springer, 2017, pp. 75–88.
[98] Q.-L. Li, J.-Y. Ma, and Y.-X. Chang, "Blockchain queue theory," in *Proc. Int. Conf. Comput. Social Netw.* Cham, Switzerland: Springer, 2018, pp. 25–40.
[99] Q.-L. Li, J.-Y. Ma, Y.-X. Chang, F.-Q. Ma, and H.-B. Yu, "Markov processes in blockchain systems," *Comput. Social Netw.*, vol. 6, no. 1, pp. 1–28, Dec. 2019.
[100] S. Ricci, E. Ferreira, D. S. Menasche, A. Ziviani, J. E. Souza, and A. B. Vieira, "Learning blockchain delays: A queueing theory approach," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 122–125, Jan. 2019.
[101] W. Zhao, S. Jin, and W. Yue, "Analysis of the average confirmation time of transactions in a blockchain system," in *Proc. Int. Conf. Queueing Theory Netw. Appl.* Cham, Switzerland: Springer, 2019, pp. 379–388.
[102] S. Geissler, T. Prantl, S. Lange, F. Wamser, and T. Hossfeld, "Discrete-time analysis of the blockchain distributed ledger technology," in *Proc. 31st Int. Teletraffic Congr. (ITC)*, Aug. 2019, pp. 130–137.
[103] M. Alaslani, F. Nawab, and B. Shihada, "Blockchain in IoT systems: End-to-end delay evaluation," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8332–8344, Oct. 2019.
[104] U. R. Krieger, M. H. Ziegler, and H. L. Cech, "Performance modeling of the consensus mechanism in a permissioned blockchain," in *Proc. Int. Conf. Comput. Netw.* Cham, Switzerland: Springer, 2019, pp. 3–17.
[105] P. Ferraro, C. King, and R. Shorten, "Distributed ledger technology for smart cities, the sharing economy, and social compliance," *IEEE Access*, vol. 6, pp. 62728–62746, 2018.
[106] P. Yuan, K. Zheng, X. Xiong, K. Zhang, and L. Lei, "Performance modeling and analysis of a hyperledger-based system using GSPN," *Comput. Commun.*, vol. 153, pp. 117–124, Mar. 2020.
[107] H. Sukhwani, N. Wang, K. S. Trivedi, and A. Rindos, "Performance modeling of hyperledger fabric (permissioned blockchain network)," in *Proc. IEEE 17th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2018, pp. 1–8.
[108] H. Zhang, C. Jin, and H. Cui, "A method to predict the performance and storage of executing contract for ethereum consortium-blockchain," in *Proc. Int. Conf. Blockchain.* Cham, Switzerland: Springer, 2018, pp. 63–74.
[109] N. Papadis, S. Borst, A. Walid, M. Grissa, and L. Tassiulas, "Stochastic models and wide-area network measurements for blockchain design and analysis," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 2546–2554.
[110] Y. Shahsavari, K. Zhang, and C. Talhi, "Performance modeling and analysis of the bitcoin inventory protocol," in *Proc. IEEE Int. Conf. Decentralized Appl. Infrastruct. (DAPPCON)*, Apr. 2019, pp. 79–88.
[111] G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Hoboken, NJ, USA: Wiley, 2006.
[112] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2014, pp. 305–319.
[113] E. Gelenbe, G. Pujolle, E. Gelenbe, and G. Pujolle, *Introduction to Queueing Networks*, vol. 2. New York, NY, USA: Wiley, 1998.
[114] M. L. Chaudhry and J. G. C. Templeton, "The queuing system M/GB/l and its ramifications," *Eur. J. Oper. Res.*, vol. 6, no. 1, pp. 56–60, Jan. 1981.
[115] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, 1999, pp. 173–186.
[116] A. Bessani, J. Sousa, and E. E. P. Alchieri, "State machine replication for the masses with BFT-SMART," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2014, pp. 355–362.
[117] "The ZILLIQA technical whitepaper," ZILLIQA Team, White Paper, Sep. 2017, p. 2019, vol. 16. Accessed: Jul. 7, 2020. [Online]. Available: https://docs.zilliqa.com/whitepaper.pdf
[118] J. Medhi, "Waiting time distribution in a Poisson queue with a general bulk service rule," *Manage. Sci.*, vol. 21, no. 7, pp. 777–782, Mar. 1975.
[119] H. Wang, J. Li, Z. Shen, and Y. Zhou, "Approximations and bounds for (n, k) fork-join queues: A linear transformation approach," in *Proc. 18th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2018, pp. 422–431.
[120] G. Ciardo, J. K. Muppala, and K. S. Trivedi "SPNP: Stochastic Petri net package," in *Proc. PNPM*, vol. 89, 1989, pp. 142–151.
[121] P. Erdos, "On random graphs," *Publicationes Mathematicae*, vol. 6, pp. 290–297, Nov. 1958.
[122] B. Bollobás and B. Béla, *Random Graphs*, no. 73. Cambridge, U.K.: Cambridge Univ. Press, 2001.

CAIXIANG FAN received the bachelor's degree from the University of Electronic Science and Technology of China, in 2012, and the M.Sc. degree from the Electrical and Computer Engineering (ECE) Department, University of Alberta, in 2019, where he is currently pursuing the Ph.D. degree in software engineering and intelligent systems. He worked as an IT Engineer at Huawei Technologies Company Ltd., China. He is currently working as a Research Assistant at the Performant and Available Computing Systems (PACS) Lab under the co-supervision of Dr. H. Khazaei and Dr. P. Musilek. His research interests include blockchain and DAG-based distributed ledger system design, performance evaluation, and modeling.

SARA GHAEMI (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering from the Amirkabir University of Technology, Tehran, Iran, in 2018. She is currently pursuing the M.S. degree in software engineering and intelligent systems with University of Alberta, Edmonton, AB, Canada. She is a Research Assistant with the University of Alberta and a Visiting Research Assistant with the Performant and Available Computing Systems (PACS) Lab, York University, Toronto, ON, Canada. Her research interests include cloud computing, distributed ledger technologies, and distributed systems.

HAMZEH KHAZAEI (Member, IEEE) received the Ph.D. degree in computer science from the University of Manitoba. He was an Assistant Professor with the University of Alberta, a Research Associate with the University of Toronto, and a Research Scientist with IBM. He is currently an Assistant Professor with the Department of Electrical Engineering and Computer Science, York University. He extended queuing theory and stochastic processes to accurately model the performance and availability of cloud computing systems at the University of Manitoba. His research interests include performance modeling, cloud computing, and engineering distributed systems.



PETR MUSILEK (Senior Member, IEEE) received the Dipl.Ing. degree (Hons.) in electrical engineering and the Ph.D. degree in cybernetics from the Military Academy, Brno, Czech Republic, in 1991 and 1995, respectively. In 1995, he was appointed as the Head of the Computer Applications Group, Institute of Informatics, Military Medical Academy, Hradec Králové, Czech Republic. From 1997 to 1999, he was a NATO Science Fellow with the Intelligent Systems Research Laboratory, University of Saskatchewan, Canada. In 1999, he joined the Department of Electrical and Computer Engineering, University of Alberta, Canada, where he is currently a Full Professor. He was the Director of Computer Engineering Program, from 2016 to 2017. He serves as an Associate Chair (Research and Planning) with the ECE Department. His research interests include artificial intelligence and energy systems. He has developed a number of innovative solutions in the areas of renewable energy systems, smart grids, wireless sensor networks, and environmental monitoring and modeling.