IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 22, 2021, accepted February 28, 2022, date of publication March 24, 2022, date of current version April 13, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3162096

Research Perspectives and Challenges of Blockchain for Data-Intensive and Resource-Constrained Devices

MUHAMMAD IMRAN[1], BIN YAO[1,2], WAQAS ALI[1], ADNAN AKHUNZADA[3], (Senior Member, IEEE), MUHAMMAD KASHIF AZHAR[1], MUHAMMAD JUNAID[4], AND UZAIR IQBAL[5]

[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Minhang, Shanghai 200240, China
[2]Hangzhou Institute of Advance Technology, Hangzhou 310024, China
[3]Faculty of Computing and Informatics, University Malaysia Sabah, Kota Kinabalu 88400, Malaysia
[4]Department of Information Technology, The University of Haripur, Haripur, Khyber Pakhtunkhwa 22620, Pakistan
[5]Faculty of Computer Sciences, National University of Computer and Emerging Sciences (NUCES) Chiniot–Faisalabad Campus, Chiniot 35400, Pakistan

Corresponding authors: Adnan Akhunzada (adnan.akhunzada@ums.edu.my) and Bin Yao (yaobin@cs.sjtu.edu.cn)

This work was supported in part by the NSFC under Grant 61832017, Grant 61922054, Grant 61872235, and Grant 61832013; in part by the National Key Research and Development Program of China under Grant 2020YFB1710200; in part by the Science and Technology Commission of Shanghai Municipality (STCSM) AI under Project 19511120300; in part by the Hangzhou Qianjiang Distinguished Expert Program; and in part by the University Malaysia Sabah.

ABSTRACT Blockchain technology in recent years has become potentially pervasive in the cryptocurrency market, thus providing tamper-proof security to decentralized transaction management systems. Structurally, the design foundation is an ideal advancement of the distributed ledger technology that maintains a set of global states across nodes. As technology expands with a higher trend towards mobile computing, the development of new applications demands understanding the current progression, especially concerning performance, data management, and storage prospects. Here, we report the principle design structure of the blockchain technology combined with the state of the arts, thus characterizing their original topological contexts. We depart from the fundamental concepts of the technology and analyze performance of the Ethereum blockchain on two devices having different computing power. Our presentation is tailored to provide a systematic review of the technology, thus facilitating their possible adoption into the new application domains like the Internet of Things (IoT). Further, we developed Debug-Bench, the first VSCode (Visual Studio Code) extension that enables benchmarking and profiling of the blockchain applications. Finally, we demonstrate several critical challenges concerning the design space of the current blockchain platforms for their implementation over resource-constrained devices.

INDEX TERMS Blockchain, blockchain data management, distributed processing, blockchain for IoT, resource-constrained devices, blockchain debugging.

I. INTRODUCTION

This is particularly true that blockchain technology has gained much popularity these days. Conceptually, it is a digital implementation of the distributed ledger technology well supported by smart contracts [1]. Nodes keep replicas of the transaction records in the form of a ledger. Transactions in a blockchain are held and processed in an ordered manner well packed into blocks. These blocks are cryptographically chained together, thus forming a blockchain. Overall, the fundamental idea behind the technology is to systematically identify and remove malicious nodes involved in the data tampering and other illegalities in settings where participants do not fully trust each other. Transactions are recorded in the same topological sequence as they appear in the block, thus guaranteeing the immutability of the distributed ledger. Fig. 1 explains the transaction life cycle of a typical blockchain application.

Blockchains in their original design [2] were permissionless (Public) by birth, allowing anyone to join and perform transactions within the network such as Bitcoin [2] and Ethereum [3], [4]. With the state replication model

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Pau.

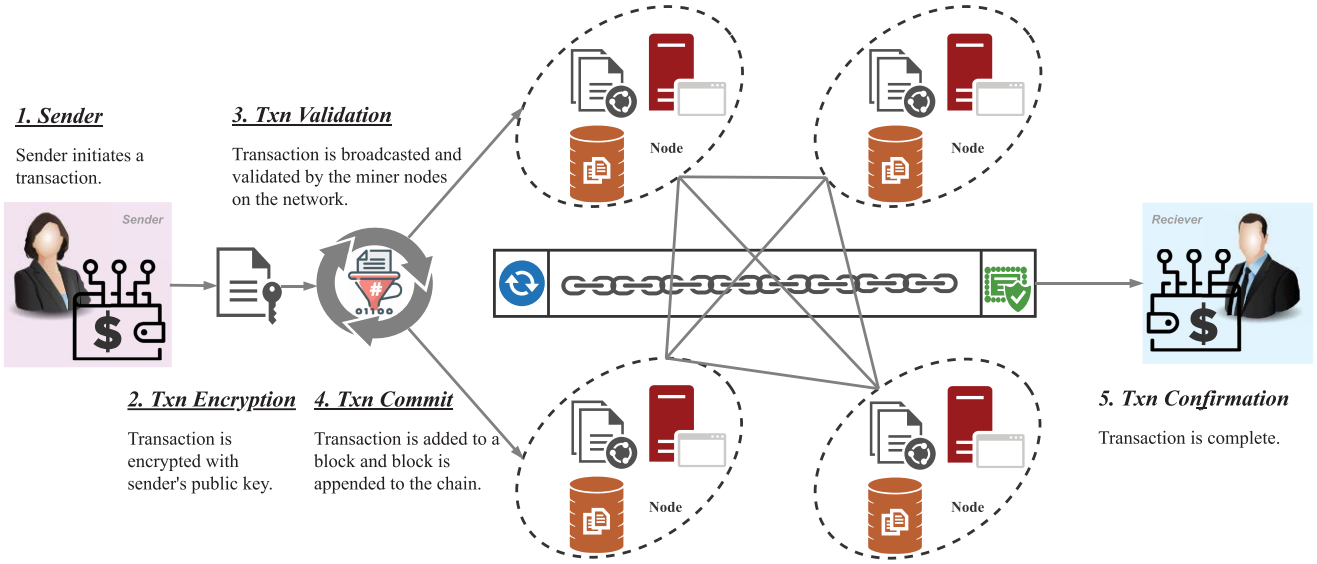

FIGURE 1. Blockchain: Transaction life-cycle.

introduced by Bitcoin, researchers made blockchain's permissioned (Private) and federated implementation possible; examples of these kinds include Hyperledger Fabric [5] and Quorum [6] blockchains. The key success behind all these types is to achieve consensus such as Proof-of-Work (PoW) [2] and make the whole process Byzantine fault-tolerant (BFT). Moreover, for the agreement to a specific ledger update, the consensus mechanism ensures agreement by all or a few (trusted in some cases) parties within the network.

Although blockchains are promising in terms of security, there are still concerns [7] about their implementation and performance for resource-constrained devices. Mining in the blockchain [2] is mainly responsible for these concerns as the miner nodes have to solve a hashing problem (aka finding signature) of varying difficulty to verify the block. Computationally, the mining process requires hardware with mighty hashing power to solve the complex combinations of a blockchain cryptographic puzzle. This requirement otherwise results in delays and latency issues. A typical block in Bitcoin takes approximately 10 minutes to complete [8]. Another concern is the lack of rich language support needed for the detailed access and representation of the transactional records. This feature is partially or even fully absent because of the key-value type of data model [9] (see Fig. 2) in major blockchains.

Recent advancements of blockchains towards securing distributed applications are also impacting the Internet of Things (IoT). However, due to smart devices' energy-performance trade-offs and resource-constrained nature, embedding blockchain into IoT is still challenging. Existing smart applications are storing tons of private data over clouds. Hackers easily breach this private data by getting into the massive traffic of IoT networks. Recently, researchers found blockchain as an opportunity to secure this private data of IoT [10]. However, structural approaches of most blockchain applications are less optimized to support low-power smart devices. Therefore, it is needed for blockchain applications to optimize current approaches to maximize their performance for IoT. Authors in recent research [11] offloaded blockchain data of smart devices with the help of edge computing.

The ongoing trend of securing distributed applications with blockchains requires a deep understanding of the technology itself. While designing new applications, developers only assess security, performance, and other technicalities on their target devices, while they partially or even fully ignore assessing computing feasibility on different devices. Several benchmarking and testing tools are available for the feasibility testing and performance evaluation of traditional applications. Blockbench [12] is one of the best tools used for performance evaluation, explicitly targeting the data processing of blockchain applications. Generally, such tools require application-specific configurations while setting up the testing environment with different IDE's (Integrated Development Environments), which is quite a complex task. While setting up our testing environment with Blockbench on Visual Studio Code (VS Code), we developed an extension Debug-Bench that facilitates deploying such configurations in a few easy steps. Details about Debug-Bench are available in the section VII.

Knowing blockchain technology, whether to modify an individual component or to build a new application as a whole, we stress one must gain a plethora of knowledge about technology in the following respects:

- What is blockchain technology, the key features, types, and contextual implementation?
- Is the technology mature enough? What blockchains are most popular to date, and what key characteristics make them different from each other.

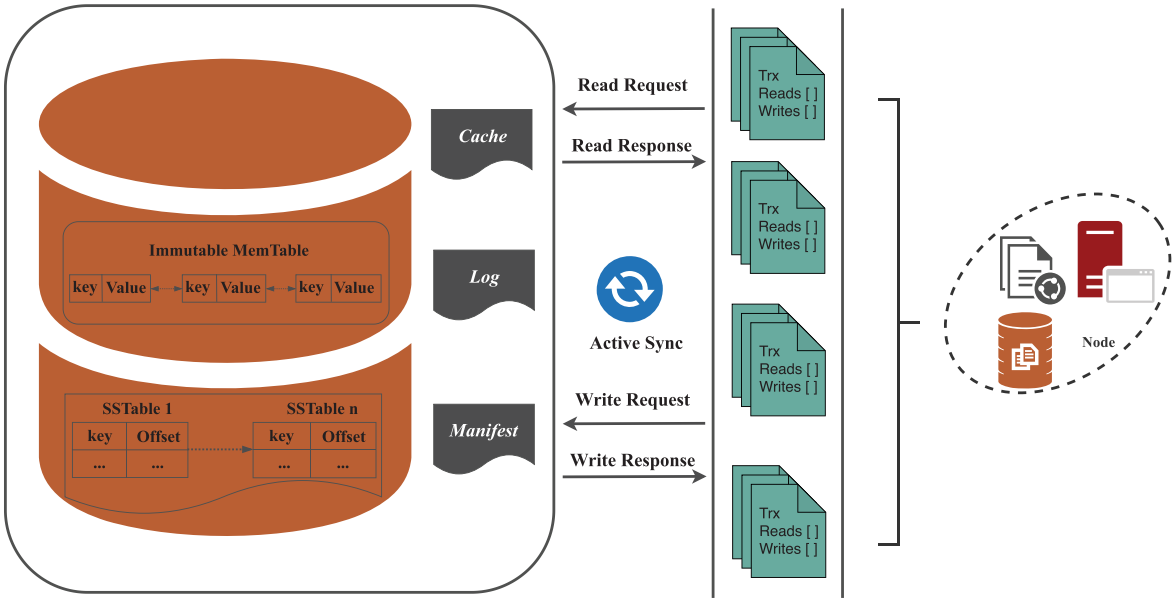

FIGURE 2. Key-value data model on the individual Node.

TABLE 1. Featuristic comparison of blockchain types.

	Public	Private	Federated/Consortium
Access	Permissionless Public Networks, Unknown participants	Permissioned, Public Networks, Known participants	Permissioned, Private LAN/WANs, Known participants
Control	De-centralized	Centralized	Partially de-centralized
Privacy	Not Guaranteed	Guaranteed	Guaranteed
Consensus	Non-deterministic	Deterministic	Deterministic
Transaction	High latency, Tamper-proof,	Efficient, Tamper-proof	Efficient, Tamper-proof
Mining	Unknown miners	Pre-selected miners	Unknown miners

- What type of data do blockchains produce? How do they manage distributed storage? What challenges do they face in data processing, especially for resource-constrained devices?
- How to evaluate the performance of the blockchain applications? Are there any evaluation frameworks so far?
- Is there any tool to facilitate debug and profiling of blockchain?

Answering the aforementioned questions and our contributions are listed below:

- We highlight the latest progress towards blockchain technology, a comparative analysis of the state-of-the-art blockchain platforms, mainly targeting data processing and storage.
- We evaluate the performance of the Ethereum blockchain for both high-power and resource-constrained devices.
- We develop Debug-Bench, a VS code extension that facilitates Blockbench [12] debugging and profiling the blockchain applications.
- We explain why current approaches used for the blockchains are less fit for resource-constrained devices (especially IoT devices).
- In the end, we share lessons learned from our study and list some research opportunities.

II. BLOCKCHAIN BASICS

This categorization of the blockchain platforms is based on join permissions, protocol execution, and the distributed ledger. See Table 1 for the parametric comparison of blockchain types.

A. PRIVATE BLOCKCHAIN

Private blockchain, sometimes called permissioned blockchain, requires permission or an authorized invitation to join the network. The user intended to join must possess its own identity over the network, or an existing user (transaction initiator, inviter, or a referring participant) validates it. Generally, an application-specific set of rules is defined on the network to set up an access control mechanism. Like the public blockchains, a consensus protocol is in place

to guarantee stable operation, which is usually deterministic. A consensus protocol is deterministic if its output remains the same for each input, provided that the initial state and the set of arguments are properly supplied. Private blockchains may adopt a consensus protocol from the available distributed consensus. Most private blockchain platforms commonly use Raft [13], Practical Byzantine fault tolerance (PBFT) [14], Zab [15], and Paxos [16]. For example, in Hyperledger protocol, a typical commit is generally completed in five stages. First, the client initiates a service request, the leader processes the request and multicast results to other nodes in the replica, replicas further execute results, and replies to the client. For the request to be valid, the results from all nodes must be the same.

Permission to join a private blockchain under standard settings is dependent on the access control mechanism. The possible rules to enter the network are gated through the existing members, an authority, or a predefined board that allows the new entrants. Besides other features, private blockchains also support smart contracts that can deal with highly complex transaction logic. When deployed under distributed settings, smart contracts ensure that unknown parties do not fully trust each other. This process involves some trusted parties acting as miners, which under private settings are usually predefined. Transactions are validated and executed by these miners based on the consensus protocol. As a result, private blockchains are relatively faster, well managed, trusted, legal, and secure than public and federated blockchains. Hyperledger fabric [17] is the popular blockchain among private blockchains.

B. PUBLIC BLOCKCHAIN

The operating structure of a public blockchain typically records transactions and the history of the states. Each participating node has its global state shared with each node on the network. A public blockchain is permissionless from the network settings, meaning everyone can join the network; however, neither participant completely trusts the other. Consequently, there is a higher chance that some nodes may act in a Byzantine manner. Bitcoin [2] is the first and most powerful public blockchain. A typical Bitcoin transaction moves states (generally digital coins) from the source address to the destination. Each participating node interested in participating in the current round broadcasts a batch of transactions to the network. Miners (nodes with powerful computing hardware) verify the transaction to be included in the next block. Mining includes a cryptographical process to compute a secure hash. Miners who successfully solve a Proof of Work (PoW) broadcast a new block to the network. Proof of Work (PoW) is the consensus protocol used initially by Bitcoin technology capable of tolerating Byzantine failure.

C. FEDERATED OR CONSORTIUM BLOCKCHAIN

A group of organizations operates federated blockchains. Nodes are only connected to a private network. No outsider can join this private network without permission. As opposed to the public blockchains, only pre-selected miners in the consensus can carry out the mining process. Other rules in the consensus protocol could be application-specific. The banks and financial institutions mainly use consortium blockchains.

D. DISTRIBUTED LEDGER

A traditional ledger systematically records an ordered list of transactions in the form of states. While in blockchains, a distributed ledger is an append-only data structure updated independently by each participating node. Similarly, the transaction recording, aggregation into blocks, and chaining are also processed independently. The distributed nature of a blockchain ledger is application-dependent; however, in Bitcoin, it is public and shared to all participants on the network, thus maintaining a computationally expensive consensus protocol only to handle the access control mechanism. Transactions recorded on the ledger are immutable, making participants confident that the records are unmodifiable. Generally, a system having distributed ledger support is first to implement a data model on the top layer of the application, such as a user-account model, a table, or a key-value model. Secondly, the ledger ownership may vary from entirely open to the public to private, strictly controlled by a single entity. Thirdly, one or more ledgers might connect in a peer-to-peer fashion.

E. DECENTRALIZED CONSENSUS

Transactions recorded to the ledger collectively represent states globally in the original topological context. Before adding a state to the blockchain ledger, all parties on the network must agree upon its identity. Otherwise, it is computationally difficult to identify who owns what without trusting anyone. To ensure a transaction to be valid, multiple parties must come to a consensus. Using a consensus mechanism is to verify that the updates being made to the ledger are valid, i.e., the network is under consensus. This is particularly important to ensure that the upcoming block being added to the chain is the most recent transaction on the network, thus lowering the risk of derailing and double-spending [18] in some cases. Note that this mechanism may be partially absent or even completely unavailable in various real-world applications. However, Bitcoin has no central authority; multiple parties come to a consensus and validate the transaction. While in this context, there is a higher probability that some nodes may act in Byzantine. Therefore, the consensus protocol must tolerate Byzantine failure. Generally, consensus protocols are classified into three bounds (Table 2). One is purely computation based in which a single randomly selected node decides the upcoming operation; Bitcoin's proof-of-work (PoW) [2] is the major example. Another is communication bound, where nodes are equally voted and go through procedural grounds to approach the consensus, such as PBFT [14]. Third comes in between these two bounds, hybrid protocols, that are aimed to address the inefficiency of either of these two bounds and

TABLE 2. Blockchain consensus protocols.

	Network Settings	Pros	Cons	Users
Proof-of-Work (PoW)	Public	Highly scalable	Slow throughput, High energy consumption	Bitcoin, Byzcoin [22], Bitcoin-NG [23]
Classic PBFT	Private	Energy Efficient, Faster than PoW, Secure	No anonymity, Works only on Permissioned settings, Communication Overhead	Hyperledger, Tendermint [24], Ripple, Scalable BFT [25], Optimistic BFT [26], Parallel BFT [27]
Proof-of-Stake (PoS)	Public	Faster processing, More decentralized, Energy Efficient	Nothing at Stake	Tendermint, DPoS [28], Peercoin [29], Ethereum Casper [30], Nxt [31]
Proof-of-Authority (PoA)	Private	Scalable, High Throughput	Fully Centralized, Only Suitable for Private Settings, Trust-for-Nothing	Parity [32]
Proof-of-Elapsed-Time (PoET)	Private	Trusted Execution	Specialized Hardware	Sawtooth [33], SGX [34]
Proof-of-Importance (PoI)	Federated	Overcome Nothing-at-Stake problem, Energy Efficient	Complex Harvesting criteria (Min 10,000 XEM Balance)	NEM [35]
Delegated-Proof-of-Stake (DPoS)	Public	Cheap transactions, Scalable, Energy Efficient	Partially Centralized	Lisk [36], EOS [37], Steem [38], BitShares [39], Ark [40]
Ripple	Federated	Faster transactions, No 51% attack, Energy Efficient	Out of date UNLs are insecure, Not decentralized enough, Not open-source	Ripple Blockchain
Stellar	Federated	More decentralized, Sharia Certified, low latency, Open-source		Stellar Network [41]
Tendermint	Private	Highly scalable, Fork Accountability, Overcome Nothing-at-Stake problem	Some latency issues	Tendermint, Casper [42], ErisDB [43]
Ouroboros	Private / Public	Highly secure, light-weight, Overcomes Grinding Vulnerability	Some Genesis and Network delay problems	Cardano [44], Ouroboros Praos [45], Ouroboros Genesis [46]

help improve the performance in addition. Prime examples of hybrid protocols are Proof-of-Authority (PoA) [19], Proof-of-Elapsed-Time (PoET) [20], and Ripple consensus [21].

F. SMART CONTRACTS

A computer code that monitors the execution and enforces an agreement in digital form between two or more parties is known as a smart contract. It triggers agreement to be executed automatically upon matching a set of predefined rules. This code enforces, facilitates, and verifies that the settlement of an automated transaction connects securely. Computationally, it is a typical form of decentralized automation similar to the stored procedures invokes upon a transaction initialized. Current blockchains with built-in

smart contracts are centered around heterogeneous transaction logics. Generally, scripts defining the transaction logics are application-specific and user-modifiable. However, these logics are compounded by the inputs, outputs, and states for cryptocurrencies. Initially, the process starts with verifying the information by checking their signatures; then, it validates the difference of the output addresses by matching with those of the inputs and finally updates states to the ledger. Current implementation approaches of some leading blockchain platforms can be distinguished based on the programming languages. Ethereum scripting functionality implements token standards that are written in Solidity or Vyper languages. ERC-20 was the first release of the token standards but was reported with some serious bugs [47]. ERC-223 and ERC-777 [48] are the next optimally more secure and efficient releases, thus resolving ERC-20 bugs.

A typical ERC-20 transaction adopts two ways to execute a token:

- `transfer()`: Send tokens to someone's address.
- `approve() + transferFrom()`: Deposit tokens to a smart contract.

  Bugs resolved in ERC-223 and ERC-777 token standards are detailed below:
- `ERC223`: It resolves the ERC20 bug by creating the transfer() function that is security-focused.
- `ERC777`: Solves transaction handling problem in ERC20 that is focused on mainstream adoption.

NEM [35] is focused explicitly on providing availability and security to cryptocurrencies. It is promising in faster execution, more available updates, better security, and lighter code. Hyperledger fabric [49] smart contract known as Chaincode, on the other hand, has been built with greater flexibility. The code is written in the Go language, supporting user-defined rules in the same language. The below snippet is a simple example of a Chaincode.

- `PutState`: A new asset is created, or updates are made to the already created ones.
- `GetState`: Pull an asset.
- `GetHistoryForKey`: Pull archived changes.
- `DelState`: Delete an asset.

Stellar smart contracts (SSC) [50] are compounded transactions that are linked and executed with several different constraints. They are implemented as agreements between multiple parties and are comparatively faster but not Turing complete [51].

G. CRYPTOGRAPHY IN CRYPTO-CURRENCY

Cryptography is one of the critical components of a cryptocurrency application. Transactions on such applications essentially demand that the information must be encrypted before transmitting to the network. Bitcoin uses public-key cryptography [52] to manage secure data transfer and access. Logically, this encryption addresses two typical problems, one to secure user identity and the other to validate ongoing transactions. Every Bitcoin transaction is assigned with a digital signature that helps keep its identity in the chain. This signature can only be generated with a valid key pair, i.e., private and public keys (ECDSA [53] based on Secp256k1 Elliptic Curve in Bitcoin by default). Just think about a typical paper cheque in the bank; a public key is similar to the account number, a private key is identical to the secret PIN, and the transaction address is the same as the cheque number. Therefore, in the Bitcoin ecosystem, a user with these keys has full access to the corresponding account. See Table 3 for a detailed overview of the security settings in major blockchain platforms.

H. HASHING

Taking some data as input, performing computation using some cryptographic algorithm, and producing a fixed-size output is called hashing. The output is always a fixed-size cryptographic hash, no matter how large or small the input data is. Hashes in blockchain help represent the current state in the global chain. Computationally, it starts from the first block so-called `Genesis`, then the hash of the previous block serves as input in generating the hash of the next block, linking blocks all the way back to `Genesis`. Bitcoin uses the Secure Hash Algorithm (SHA) and the RACE Integrity Primitives Evaluation Message Digest (RIPEMD) algorithm [54] for hashing, specifically RIPEMD-160 and SHA-256. Since each block in a blockchain contains thousands of transactions; therefore, it is not feasible to store all of the meta in a block. Bitcoin uses the hash of the Merkle Tree to cut down data size and the time required to find out whether a particular transaction belongs to the current block or not.

I. MINING

Mining is the key feature of blockchain introduced by Bitcoin blockchain. Logically, Mining is not only the process by which a new coin is generated but also serves to secure applications from attacks like frauds and double-spending. This is how Bitcoin has outsourced processing power in exchange for the reward for performing validation that whether a particular transaction in the chain is valid or not. Recall the subsequent sections where we discussed cryptography that the second and most critical use of hashing in Bitcoin is transaction validation (so-called `Mining`). To do so, Bitcoin broadcasts each block to the network where `Miners` perform Mining by solving a cryptographic puzzle in exchange for a small fee. This puzzle is cryptographically hashed with a higher level of difficulty that requires a high processing power node (computing machine) to get it solved.

J. BYZANTINE VS NON-BYZANTINE

Byzantine problem states that the two nodes on a distributed network are acting maliciously. The Mining process at its core entirely relies on the Miners (validating nodes) in the global network. Any of these nodes may act Byzantine and mislead others against the protocol. Bitcoin being the first to face, has successfully addressed this problem by implementing a consensus protocol (PoW). Several

TABLE 3. Security settings of major blockchain platforms.

	Access	Mining	Fault-Tolerance	Hashing	Immutability	Privacy
Hyp. fabric	Permissioned	Peer Nodes (Endorsement Policies)	Byzantine with orderer service	SHA-256	Guaranteed	Guaranteed within Channel
Ethereum	Permissionless	Public Nodes	Byzantine	Keccak-256, Merkle	Guaranteed	Not Guaranteed
BigchainDB	Permissionless	Cluster Nodes	Byzantine with Tendermint	SHA-256, Merkle	Guaranteed	Not Guaranteed
Quorum	Permissioned	Network Members	Byzantine with GoEthereum	PGP	Guaranteed	Guaranteed
R3 Corda	Permissioned	Trusted Nodes	Byzantine with Notary Services	SHA-256, Merkle, ECC	Guaranteed	Guaranteed
MultiChain	Permissioned	Trusted Nodes	Round robin	SHA-256	Guaranteed	Guaranteed
IOTA	Permissionless	Anyone on network	Byzantine	Curl-p	Guaranteed	Not Guaranteed
Ripple	Permissionless	Trusted Validators	Byzantine	SHA-512	Guaranteed	Partially Guaranteed
NEM	Hybrid	Any Nodes (Harvesters)	Byzantine	SHA3-256	Guaranteed	Not Guaranteed
Monax	Permissioned	Voting based	Byzantine	Keccak-256	Guaranteed	Not Guaranteed

consensus protocols have been introduced to date (see Table 2), which is sufficiently resolved. PBFT has solved the Byzantine problem for cryptocurrencies and across other major blockchain-based implementations. PoW, PoS, PoET are other popular consensus protocols to date. Many blockchain applications tolerate Byzantine failure either by implementing a fault-tolerant consensus protocol or re-adjusting communication parameters in public or private settings. Likewise, some blockchain platforms have entirely separated their consensus layer and handed it over to a single trusted third party. However, this strategy is totally a one-point failure and has a comparatively higher implementation cost. Such blockchain platforms are said to be of the Non-Byzantine type. Multichain [55] and Parity rely on more than one trusted third party. Openchain [56] relies on a single trusted third party called `Validator`. Overall, the Non-Byzantine approach is also practically efficient for some platforms like R3 Corda is using a set of trusted third parties known as `Notaries`. Similarly, Hyperledger (v1.0) switched its consensus components to Kafka [57], formerly Hyperledger Kafka.

K. ON-CHAIN VS OFF-CHAIN

Chaining transactions into a block, executing consensus protocols, performing validation, and storing detailed information on the chain is time-consuming and computationally expensive. The concept of On-Chain refers to that everything is being processed on the same blockchain. Being selective towards overall scalability under the on-chain paradigm, Bitcoin is limited to 7 transactions per second [58], while Ethereum supports up to approximately 15 transactions per second (TPS) [59]. Further, confidentiality and privacy under on-chain settings are also not guaranteed. Here comes the concept of off-chaining, where some features from the underlying technology are partially compromised or even entirely not manifested, which is logically the same as locking a process down at a specific point and off-loading its computation along with the data to somewhere else. Sidechains [60] are also centered on this concept. The off-loading process may involve one or more trusted intermediary(ies) for various purposes, i.e., validating transactions, storing data, multiparty verifications, privacy, confidentiality, and even switching to other crypto-currency platforms. Off-chaining is beneficial to some applications such as financial institutions and banks, where they want to incorporate manual settings between the process. Several applications on top of a specific blockchain platform are introduced, which partially manifest their fundamental layers underlying the technology.

L. SQL VS NoSQL

SQL databases are mainly used where it is required to store highly structured (relational) data with predefined schemas. On the other hand, NoSQL is used for the unstructured (non-relational) data with usually a dynamic kind of schema.

TABLE 4. Data model of major blockchain platforms. Terms and abbreviations used in table are: KV:Key Value, FB:File Based, AB:Account Based, BB:Block Based, DL:Distributed Ledger.

	Data Model	Storage	State Management	Chain Structure	Advance Queries	Analytics
Hyp. fabric	NoSQL, KV	RocksDB	Storage Engine.	Bucket-Merkle	No	No
Ethereum	NoSQL, KV	LevelDB	Stack,in-memory	Merkle	No	No
BigchainDB	SQL, FB	MongoDB	Storage Engine	Merkle	Yes	No
Quorum	NoSQL, State DB					
R3 Corda	SQL, FB	H2	in-memory	B-Tree	Partial	No
MultiChain	NoSQL, FB	LevelDB	in-memory	Merkle, LSM	No	No
IOTA	NoSQL, KV	DAG D Ledger	in-memory	DAG	No	No
Ripple	NoSQL, AB	XRP Ledger	in-memory	XRP	No	No
NEM	SQL, BB	RocksDB	in-memory	Merkle	Partial	No
Monax	NoSQL, AB	ErisDB/LevelDB	Stack,in-memory	Merkle	No	No

According to the CAP theorem [61], strictly structured SQL databases cannot be truly distributed where one has to choose a feature between availability and consistency. However, NoSQL databases overcome this problem by replacing the term consistency with eventual consistency [62]. NoSQL databases with eventual consistency have become successfully efficient in storing decentralized data of the blockchain applications. There are several mature realizations with such implementation, i.e., BigchainDB, MongoDB [63], RethinkDB [64] and so on. This is particularly true that the NoSQL databases are scalable, fault-tolerant and fast but have serious limitations (in some cases) in their ability to tolerate Byzantine failures. BigchainDB being the major adopter of the NoSQL paradigm suffers from the same limitation. Nodes at such a platform completely trust each other, meaning that a single malicious node can destroy the entire database.

III. BLOCKCHAIN DATA MANAGEMENT AND ANALYTICS

Current blockchain approaches dealing with high-dimensional security problems have proven themselves promising. However, they have serious limitations in their ability to scale with the ever-growing data, thus resulting in a problem called the storage bloating problem [65]. One entire blockchain in the original Bitcoin takes 66 GB, and this number is increasing by 0.1 GB per day [65]. The estimated size will be 40 TB in 20 years. One choice is to adopt a simple key-value store that is highly scalable in providing mild storage but fails to support rich queries and lacks ACID properties. Another choice is relational storage that supports SQL and ACID properties with structured data storage. In between are the applications that make other tradeoffs between the data model, semantics and performance. Table 4 provides detailed insights into the data models of major blockchain platforms. One challenge in securing blockchain applications requires the ledger to be tamper-proof. Therefore, nodes must broadcast the most recent ledger copy to the network each time a transaction is committed. This consequently increases network traffic that further may introduce latency and communication overheads. Such a model demonstrates the crucial importance of its implementation over resource-constrained devices like IoT. Today's blockchain platforms reside on the top layer of the key-value store (see Fig. 2). They are promising in security but generally less efficient in supporting analytical and rich query processing. Being selective towards the alternatives, a recent attempt has been made by Sheng [66]. They designed ForkBase, an efficient storage engine to support blockchains and forkable applications. However, they have not evaluated its performance on resource-constrained devices (e.g., devices equipped with low-power processors like ARM).

Another challenge in scaling blockchains is to minimize communications and limit data transfer. One solution is to design a well-structured database that could support flexible data types. Such a design may offer partial support to the analytical queries. Applications like ForkBase may not best serve this purpose. Another solution is to employ some classification techniques on the raw data to extract and transfer only useful information. However, this approach has other limitations from the implementation prospect where network settings are public. It is essential to transfer complete information to perform block mining in such applications. This makes the implementation resource intensive, especially for the devices equipped with low-power processors.

IV. PERFORMANCE BENCHMARKS

We evaluate Ethereum (Geth v1.8.27-stable) blockchain to quantitatively analyze design and performance gaps as a data processing platform. This practical demonstration is to help understand what we have theoretically explained in previous sections and identify real-time performance gaps in Ethereum blockchain for high-power and resource-constrained devices. This is of particular importance to mention here that to the best of our knowledge, none of the current blockchains are practically implemented on resource-constrained devices, e.g., devices equipped with ARM processors. We chose Ethereum for our testing because of its universality and the specialty to support all types of implementations, e.g., public and private (in some cases where privacy is added on top layers of the framework, for example, Hydrachain [67] and Quorum [6]). Ethereum supports both contract and non-contract types of user-defined accounts that are directly read into states. These states are partially stored in cache memory and fully stored on disk-based storage (a key-value store). On the other hand, other major blockchains have limits in these respects. Therefore, being a universal platform, we think Ethereum can provide better insights into design and performance derivatives. We use BLOCKBENCH [12], a benchmarking framework that contains all necessary data processing workloads to analyze lower layers of the blockchain applications. We use microbenchmarks to determine the consensus layer's overall performance, which further involves operations by the data model layer and the execution layer, respectively. We use `Analytics` and `IOHeavy` workloads in our testing where the former is similar to the OLAP (Online Analytical Processing) workload while the latter evaluates the Input-output (IO) performance by sending random reads and writes to the local states in bulk. All experiments are performed on two machines, first on the Linux-based server with 2 GHz 24 core Intel CPU and 190G memory, and second on the Linux-based server with 2.6 GHz 8 core ARM CPU and 16G memory. Each geth (Go-Ethereum) execution was initialized to use a maximum of 4 peer nodes in the commodity cluster.

A. CPUHeavy WORKLOADS

In order to evaluate the execution layer, we initialized CPUHeavy smart contract to sort an integer array of 1 million numbers. We used four miner nodes and invoked smart contracts for each miner node one by one. We measured CPU workloads (core, crypto, consensus, runtime, runtime.memmove) and latency by repeating each `Run` three times under the same configurations. A results-based comparison for both Intel and ARM servers with the varying number of miners is shown in Fig. 3, Fig. 4, and Fig. 5. We noticed varying workloads and the execution time for each `Run` while keeping execution-related configurations the same. For example, in Fig. 3, the CPU usage on the Intel server compared to ARM in the core module is higher for one and two miner nodes, while it is lower for 3 and 4 miner nodes, respectively. On the other hand, in the third execution Fig. 5, the Intel server has higher and fluctuating margins, while the ARM has lower and consistent clock time. There is also a significant margin in all modules among the three executions having the same configuration and input sizes. Likewise, the latency for 3 and 4 miner nodes on the Intel server in all three `Runs` is higher than the ARM. Overall, the results have shown a similar trend for all modules except latency in all three `Runs` under CPUHeavy workloads with varying miner nodes, see Fig. 3, Fig. 4, and Fig. 5. Upon further inspecting results in plots for each `Run` with a varying number of miner nodes, we observed a decline in CPU usage for the core module on the ARM server and a notably significant raise in the latency on Intel. This indicates that Ethereum incurs module-specific overheads for both Intel and ARM servers which we observed is associated with the execution of high-level EVM byte code. Notably, a five times higher latency on an Intel server with an input size of 4 miner nodes while sorting 1 million integer numbers is quite unusual in all three `Runs`. However, It runs out of memory and continuously hangs while sorting 10 million or more numbers while keeping other settings the same.

B. IOHeavy WORKLOADS

We evaluate the data model layer by deploying an IOHeavy smart contract on Geth. All the executions are based on some read and write operations performed into key-value tuples since Ethereum uses LevelDB, a key-value store. Both operations execute queries with 10000 key-values written or scanned into the database. Fig. 6 shows results with varying input sizes for Intel and ARM servers. We observed a significant unusual increase in latency of both read and write operations on ARM servers for all four `runs`. It incurs a higher latency on the ARM server for each input size (No. of miners) because it is associated with the process of validation overheads in block mining. Similarly, We observed a higher memory usage on the ARM server than on the Intel. The core execution for the INTEL Read operation with one miner node, as shown in Fig. 6a, takes almost 50% longer than ARM. However, it is stable in another Run with two miner nodes. While in Fig. 6d, We observed an opposite behavior of the write operation on the `run` with two miner nodes where ARM execution takes 60% longer than INTEL. Again, this behavior is due to the execution of high-level EVM byte code in the core part, similar to the results of CPUHeavy workloads explained in the previous section. Overall, we noticed better results on the Intel server for IOHeavy workloads. Fig. 6 shows the detailed results.

C. ANALYTICS WORKLOADS

We deployed analytics workloads on the data model layer by executing three types of analytical queries mentioned below:

q1 - total transaction value from block i to e
q2 - largest transaction value from block i to e
q3 - largest account balance from block i to e

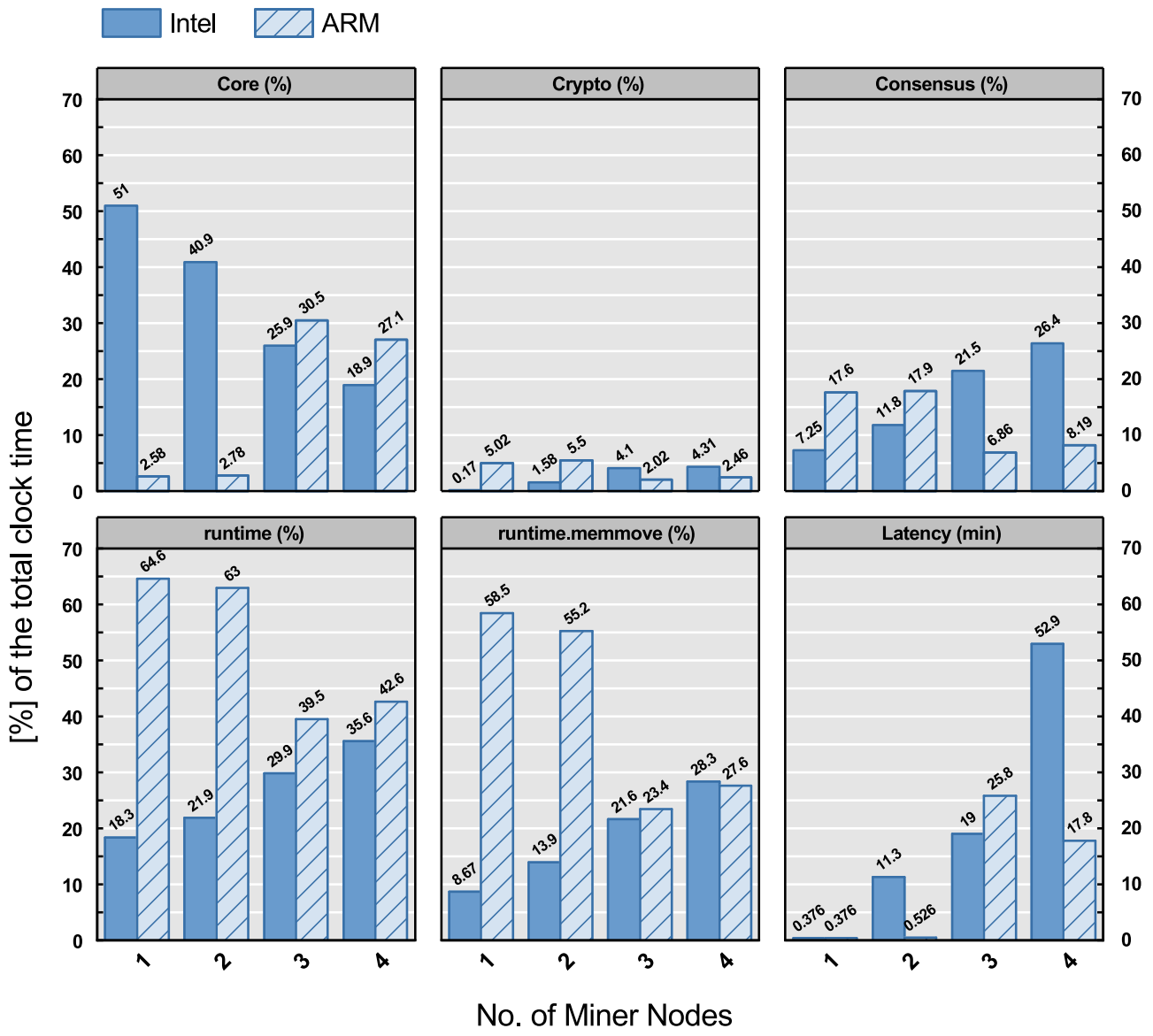

FIGURE 3. Performance comparison of Ethereum CPUHeavy sorting workloads with 4 miner nodes [1st Execution].

We initialized the geth account with 10,000 blocks, each block packed with three transactions. Generating three transactions per block is already defined in the BLOCKBENCH. However, we executed the queries mentioned above one by one, first with one miner node and then with two miner nodes. We observed a notably high latency while executing q1 with one miner node on the ARM server and fluctuating results in CPU usage. Fig. 7a and Fig. 7b shows the results. However, we observed a less significant difference in the results of queries q2 and q3.

V. BLOCKCHAIN FOR IoT

Because blockchains are secure and flexible, their application in many areas is looking for a major breakdown that could somehow ease its adoption. A most recent trend of their adoption has been observed towards IoT. IoT applications produce and transfer tons of privacy-sensitive information over the network each second, thus tempting targets of various attacks. Deploying such applications into remote areas requires devices to be lightweight. Such devices are equipped with low energy and minimum storage. These constraints make conventional security methods computationally expensive on low-power devices. Blockchain overcomes the challenge mentioned above; however, the implementation is not straightforward due to the complex computation required for block mining. Recent attempts [68]–[71] of implementing blockchain into IoT are largely organized by making considerable changes in the design space.

While conceptualizing sensitive information of IoT mainly in the smart home context, we observed some critical security limitations. The underlying transaction information is exposed to each node in the original blockchain design, where a block is broadcasted to each node for verification, thus rendering significant privacy concerns. Such a data structure

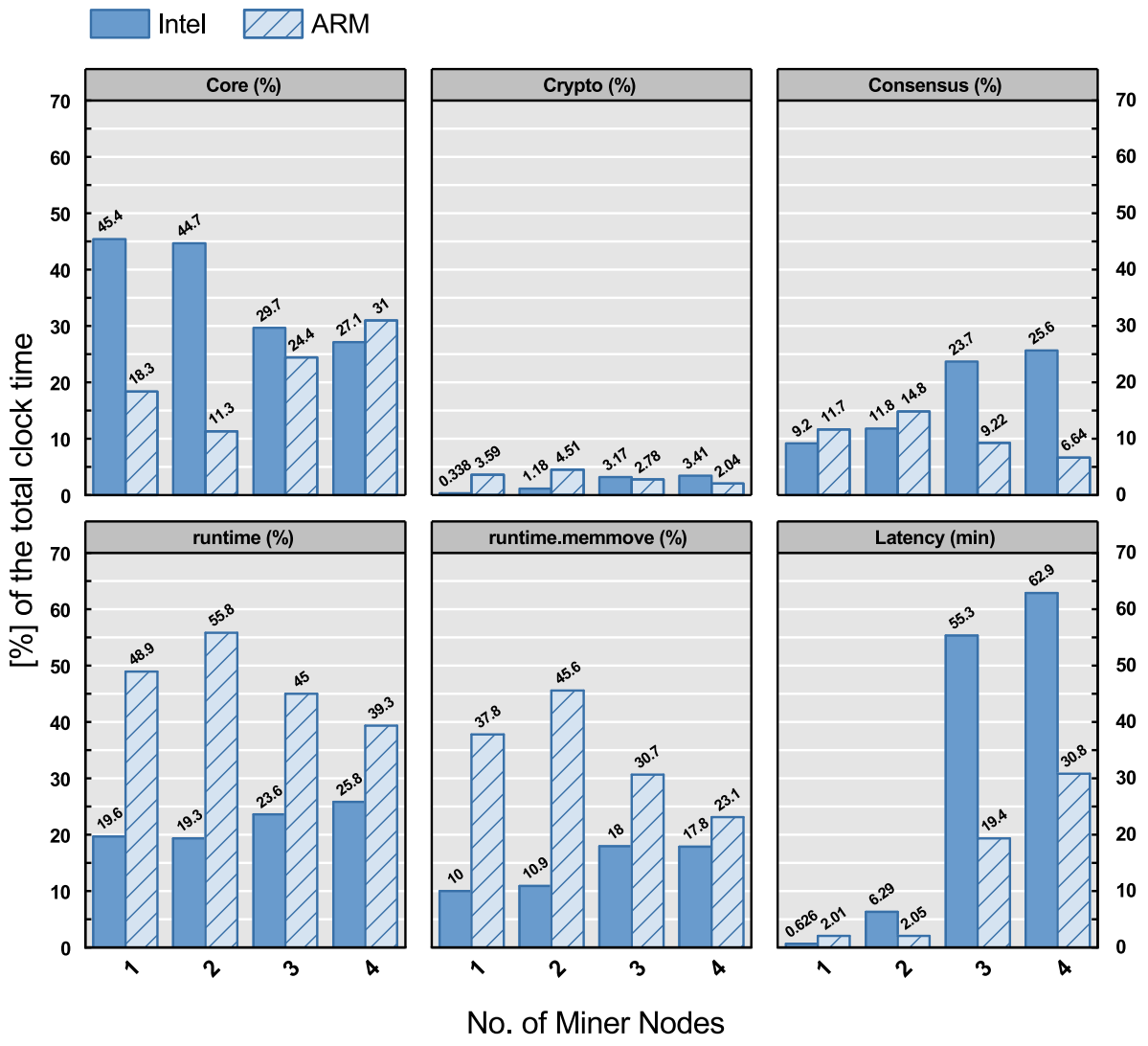

FIGURE 4. Performance comparison of Ethereum CPUHeavy sorting workloads with 4 miner nodes [2nd Execution].

is only suitable for the applications where blockchain settings are public, e.g., Bitcoin and Hyperledger. Authors revealed that today's IoT devices lack essential security considerations that are demonstrated in [69], [72] and [73]. Once again, the challenge is to design an application-agnostic architecture and should cope with both blockchain and IoT settings.

A. WHY TRADITIONAL BLOCKCHAINS ARE LESS SUITABLE FOR IoT

Conventional blockchain approaches on resource-constrained devices are still insecure, especially when IoT devices are mobile. While reviewing and analyzing the latest solutions in the previous sections, we observed some vulnerabilities that are still challenging in their implementation.

A simple vulnerability attack could compromise device identification based on similar data patterns in IoT. Attackers analyze user patterns in the sensed data to identify the device, thus compromising user privacy. One solution is mentioned here [70]; another is to analyze data records locally and transfer only meaningful information to the network. However, this solution requires the data model to be well structured to handle the update requests wisely.

Since blockchains are immutable, therefore, transactions are permanently stored in the database, thus potentially increasing the data size, which is not suitable because of the limited storage capacity of the IoT devices. Additionally, hackers may easily breach the over-flooded data.

Another significant limitation is the mining process (solving cryptographic puzzles) that requires much computation at the local node. Once again, it is reasonably infeasible for resource-constrained devices as they are equipped with low-power processors (e.g., ARM).

Blockchain's approach to processing the information on the chain opens many challenges for the IoT. We found that the major blockchain platforms use key-value stores to store data records because of their lightweight implementation, thus compromising rich query support. Traditional SQL

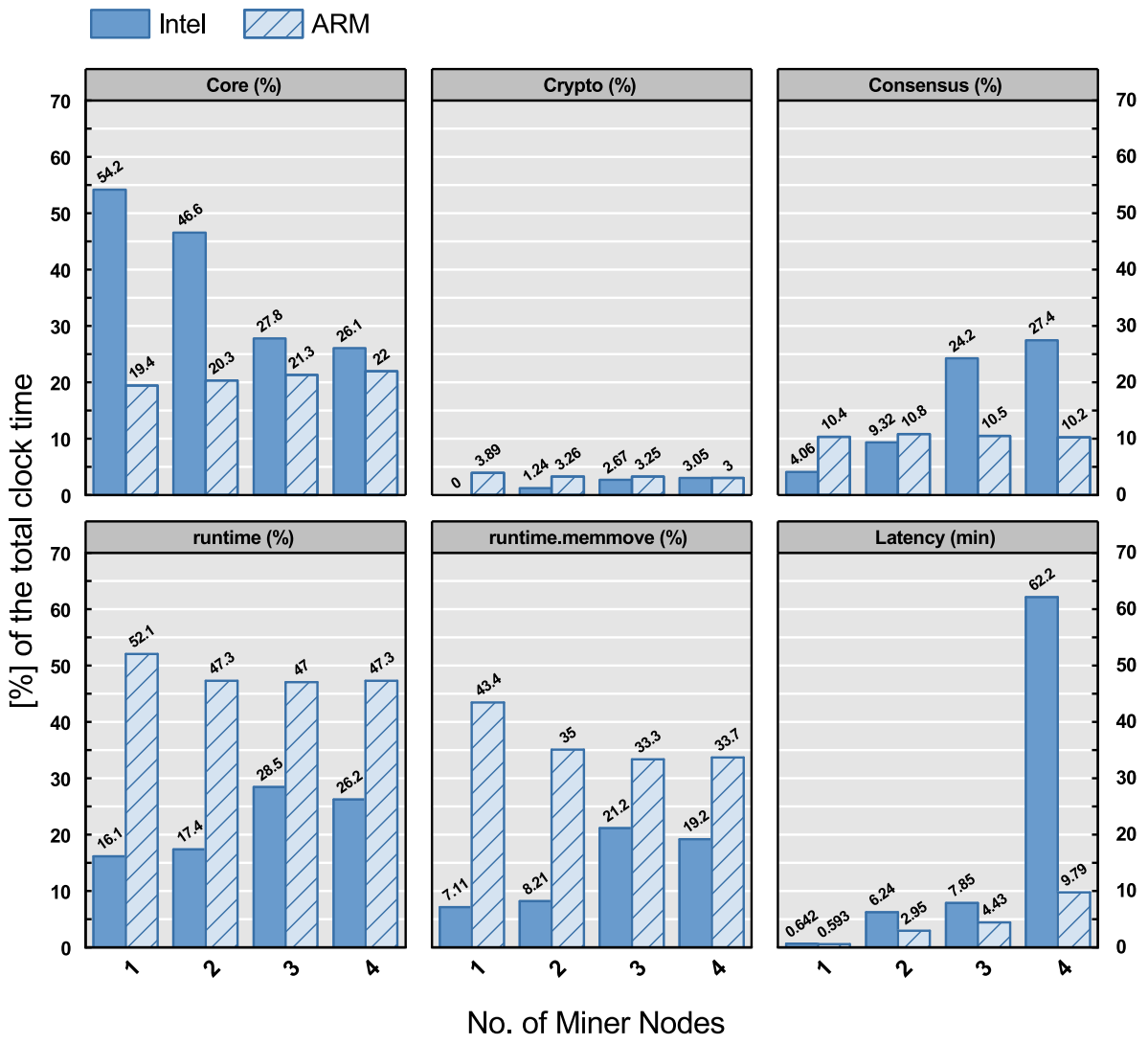

FIGURE 5. Performance comparison of Ethereum CPUHeavy sorting workloads with 4 miner nodes [3rd Execution].

databases in this context are not feasible because of their large size. On the other hand, IoT requires data models to be efficient in terms of rich queries and analytics support. One solution is to go for off-chain storage, which could lead to user deanonymization, thus referring to blockchain technicalities. However, study [68] shows that this limitation can still be dealt with using other approaches.

Cloud applications generally accommodate different networks receiving and transferring hundreds of thousands of records each second. This accommodation requires regular support to take topological and structural insights into account, which we usually found discriminating towards blockchain settings. None of the current blockchain platforms fully support heterogeneous networks due to the aforementioned security constraints to the best of our knowledge.

VI. DISCUSSION

In this section, we share our experiences learned during this study. We first demonstrate the technological insights that have been under focus in recent years. We discuss the pros and cons of the design principles currently being followed by most blockchain platforms. We then list some research opportunities for future blockchains, specifically from the data processing perspective and their integration into resource-constrained devices like IoT.

A. DESIGN AND PERFORMANCE TRADE-OFFS

Since researchers discovered blockchains promising in terms of security, many technology components are still demanding major optimization. Even under the security context, serious flaws were identified and fixed after the giant Bitcoin was exposed to some major attacks [74], [75]. In this context, we present performance statistics of some major blockchain platforms in Table 5. We calculated system maturity scores by assessing system components according to the KPMG blockchain maturity model [76], which measures blockchain risk areas against the CMMI maturity model [77]. A flexible system design lets users develop applications according to their specifications and standards.

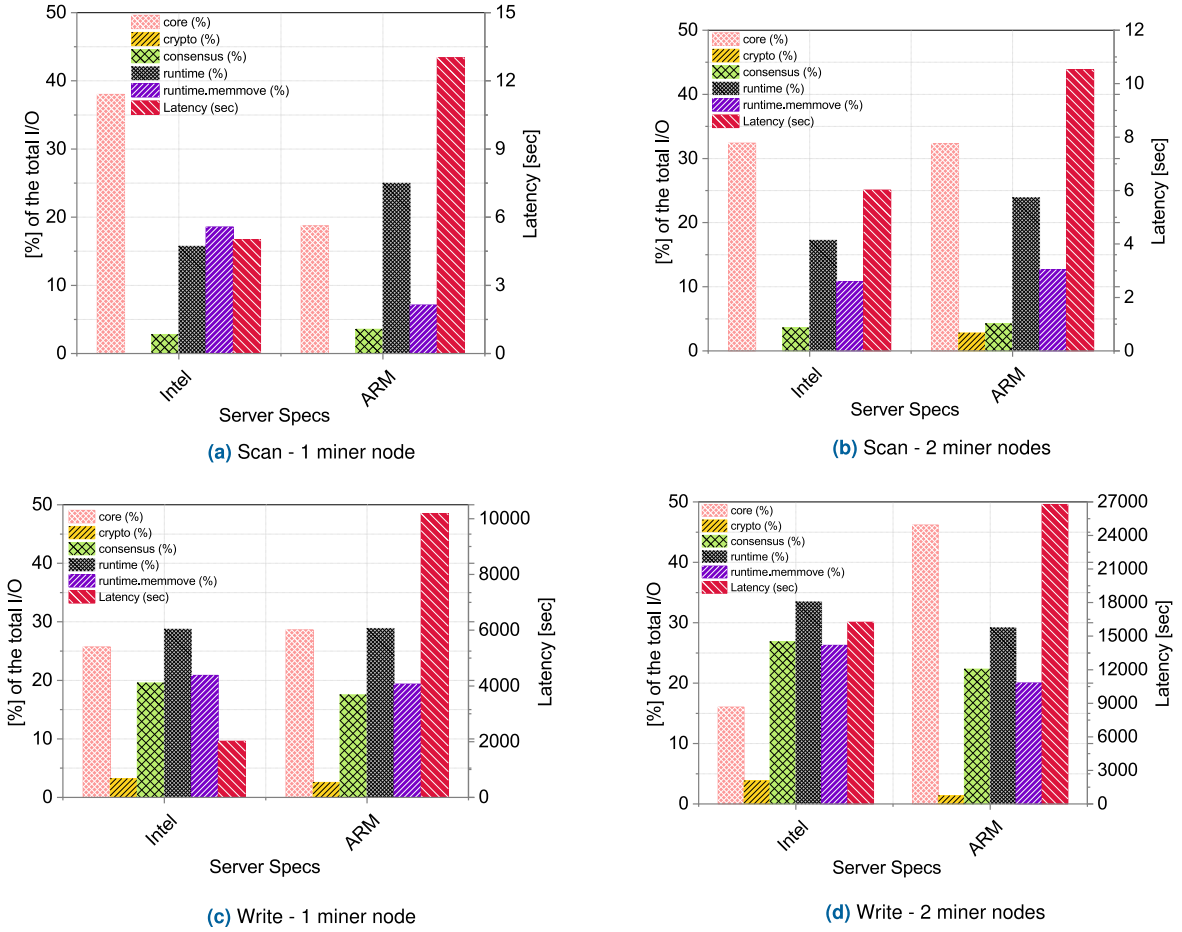

FIGURE 6. Performance comparison of Ethereum IOHeavy workloads.

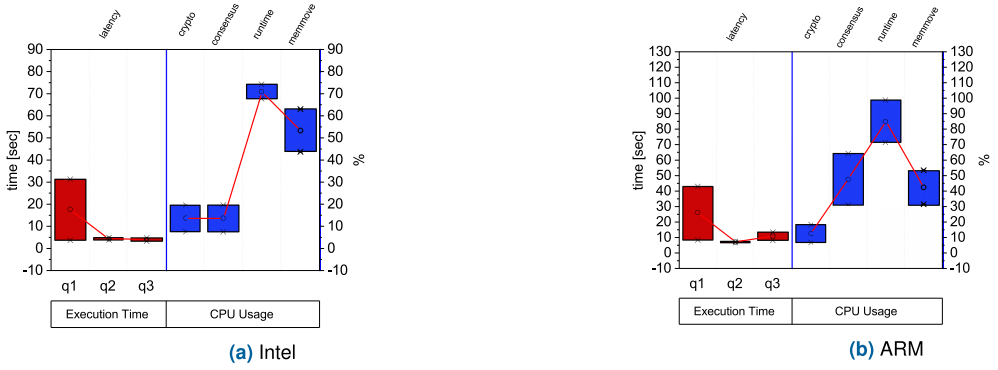

FIGURE 7. Performance comparison of Ethereum analytics workloads.

After Bitcoin introduced its simple state replication machine, the significant focus of the technology diverted towards the development of private blockchains. The modules discovered under private settings [78] differ in their structural and topological abstraction because of the public settings, which typically remain unchanged (see Table 6). These modules are largely responsible for the application-specific dependencies, thus uncovering scalability concerns under private settings. Being selective towards computational requirements, today's blockchains require highly specialized hardware to perform the block verification (or mining in some PoW blockchains). We see this requirement as a huge barrier for the blockchains (not all types of consensus) in their implementation on resource-constrained devices such as devices with low-power processors. One extreme is the applications with high-security requirements where current blockchains perfectly fit themselves. Another extreme is the applications demanding both security and high performance,

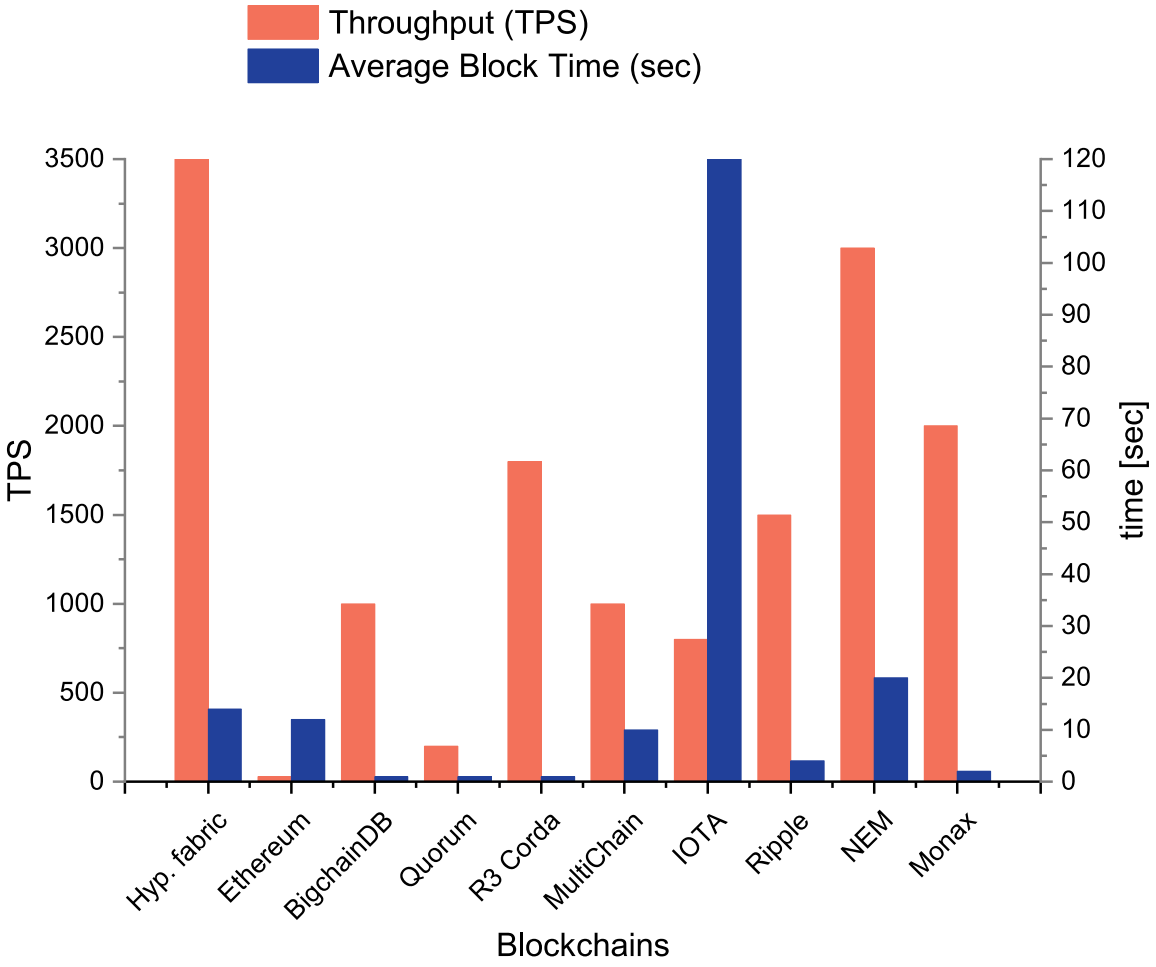

FIGURE 8. Performance statistics of major blockchain platforms.

blockchains for such applications require specialized hardware. However, applications like IoT less support specialized hardware because of various constraints described here [68], [79], [80].

Major blockchains to date implicitly follow traditional design principles that do not cope with the heterogeneous types of integrations. However, a few of them [5], [21], [35], [55], [81] allow partial customizations like user-defined settings for block size, block time, and even building a business-specific application on the top layer of the framework. However, they are less flexible in supporting modifications in the design invariants. Keeping structural and design specifications fixed has benefits, especially in public network settings. We also stress the importance of the data processing and storage approaches, which are currently limited to key-value stores [9], [82], [83], thus compromising analytics and advanced querying support. BigchainDB and ForkBase are some of the latest attempts in this context. Table 5 and Fig. 8 show detailed insights into the performance statistics of major blockchain platforms. Computationally, we conclude that blockchain technology in both private and public settings is still expensive for low-power devices.

1) SCALABILITY

A quest to achieve scalability for public and private blockchains continues. Even with the satisfactory improvements in performance, especially for the private blockchains, current approaches are less incapable of dealing with the large-scale data workloads. Additionally, the block mining and consensus mechanism consume tremendous amounts of energy, thus making their implementation infeasible for resource-constrained devices.

2) STORAGE AND DATABASES

Generally, today's blockchain platforms reside on top of the key-value stores like LevelDB and CouchDB. These databases are lightweight but less capable of supporting indexing and advanced queries. Further, such platforms are less suitable for cross-application integration because they produce a wide range of raw data. Therefore, new approaches are needed to cope with multiple types of data stores efficiently.

3) ACID BASED DATABASE TRANSACTIONS

Like the traditional databases where ACID properties help resolve many problems in the relational databases and

TABLE 5. Performance statistics of major blockchain platforms. Terms and abbreviations used in table are: Avg:Average, TP:Throughput, TPS:Transactions per second, DF:Design Flexibility, UD:User-Defined, Cl:Cluster.

	Nodes	Avg TP (TPS)	Avg Block Time (Sec)	Maturity (10)	Block Weight(Mb)	DF (10)	Block Size (Trx)
Hyp fabric	26	3500	Instant	2	UD (Default 0.5-98)	4	UD (Default 30)
Ethereum	8192	27	12	8	21	6	106
BigchainDB	4/Cl	1000000	Big	-	16	-	1000
Quorum	4/Cl	150-1650	Instant	6	1	6	750
R3 Corda	UD	1000-1800	Instant	2	1	6	1000
MultiChain	UD	600-1000	UD	6	1-1024	7	UD
IOTA	99	500-800	60-120	4	No-limit	6	No-limit
Ripple XRP	1040	1500	4	5	UD	6	UD
NEM	422	3000	20	4	UD	7	UD
Monax	UD	2000	Instant	6	UD	6	UD

TABLE 6. Structural comparison of blockchain platforms. Terms and abbreviations used in table are: Gen:General, Apps:Applications, Prog:Programmable, Srv:Services, DA:Digital Assess, Fin:Financial, Ch:Channel.

	Scope	Design	Consensus	Language	Trx Ch.	Application
Hyp. fabric	Private	Flexible	Multiple, Apps-based	Golang, Java	On-chain	Gen. Apps
Ethereum	Public	Prog, Flexible	Proof-of-Work (PoW)	Bytecode, Golang, C++, Solidity, Rust	On-chain	Gen. Apps
BigchainDB	Public, Private	Fixed, Prog	Tendermint BFT	JS-Crypto, Python	On-chain	Gen. Apps
Quorum	Private	Flexible	QuorumChain, RAFT, IBF	Golang	On-chain	Gen. Apps
R3 Corda	Private	Fixed	Notary Srv, Raft	Java, Kotlin	On-chain	DA,B2B Apps
MultiChain	Private	Flexible	Trusted Nodes	C++	On-chain	DA,Fin Apps
IOTA	Public	Fixed	Tangle, PoW	Java, C++, Rust, Golang	On-chain	DA,Fin Apps
Ripple	Public	Fixed	Ripple	C++	On-chain	DA,Fin Apps
NEM	Public, Private	Flexible	PoI	Java, C++	On-chain	Cryptos, DA
Monax	Private	Flexible	Tendermint	Solidity	On-chain	Gen. Apps

take full advantage of the SQL and advanced querying capabilities, blockchain requires the same treatment. Sadly, major blockchains only partially support ACID properties for the database transactions. They partially practice simple concurrency control or even not. The key concern here is the scenario when multiple transactions want to access the same data; however, the transaction getting endorsed first has the right to access that particular record first while letting others wait for their turn. We report two problems here, first is the latency that all subsequent transactions wait for their turn. The longer the queue higher the latency. Second, transactions waiting in the queue are not sure whether that particular record will be available on their turn or not, thus turning their wait into waste. Hence, a significant challenge is to design a concurrency control mechanism that could handle decentralized transaction management by guaranteeing ACID properties fully.

4) ANALYTICS-READY EXECUTION ENGINE

Blockchain-enabled applications are entirely transaction repository platforms; therefore, one possible way would be to run analytics directly on the raw data. Logically, we stress that the analytics must be executed in parallel with the execution engine immediately after the application layer. In this sense, an input-reader like [84], Hadoop [85] or Flink [86] could do the batch analytics needed. Edge analytics [87] would be needed for blockchain-based IoT or smart home applications. Analytics in blockchains is also important in where cross-chain or off-chain data integration is needed.

5) CROSS-CHAIN DATA INTEGRATION SUPPORT

It is worth noting that today's blockchain applications work independently of the superseded systems, thus making cross-chain sharing infeasible as different organizations use

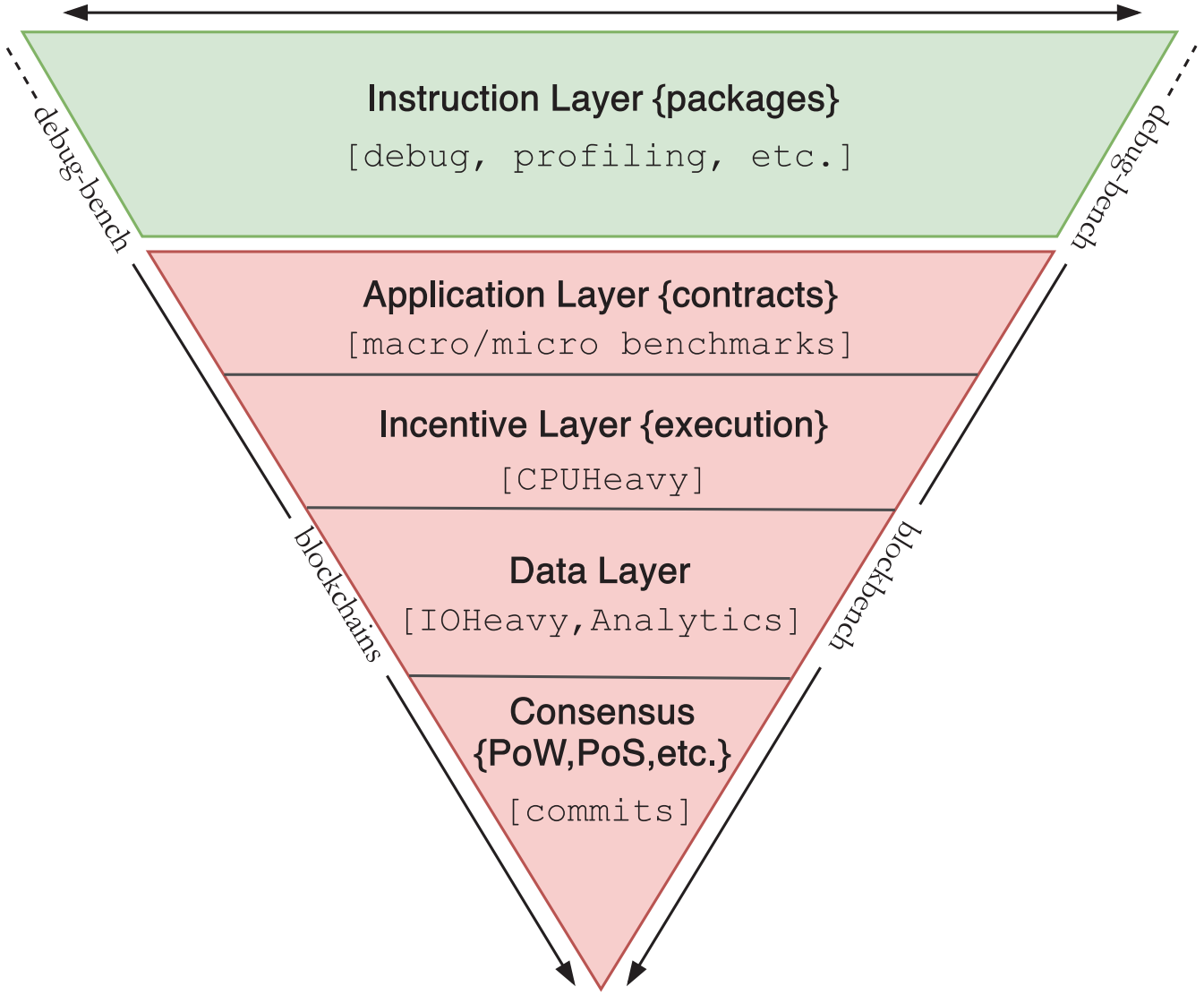

FIGURE 9. Debug-Bench: Instruction layer.

different ledgers. Not putting forth important properties like safety and liveness across multiple ledgers increases the risk of sidechain failures in blockchains. A perfect sidechain construction for PoS and PoW based sidechains is provided here [88], [89]. Further, an organization developing a new blockchain application would require integrating with their existing systems to get previous data records. A higher probability of data overlapping and inconsistency may arise in such cases and cases where multiple parties join a single blockchain network without intermediaries.

VII. DEBUG-BENCH

Thanks to the BLOCKBENCH as the first successful evaluation framework for blockchain applications. It analyzes blockchain applications by enabling workload-specific simple APIs, thus providing detailed benchmarked analysis based on inputs with varying types of workloads. The evaluation process with BLOCKBENCH starts with the initialization of smart contracts over the application layer and works in parallel with the contract execution process. The layered structure is explained in Fig. 9.

In our experiments, we used BLOCKBENCH to evaluate the performance of the Ethereum blockchain; results can be seen in section III. We observed two complexities while setting up BLOCKBENCH's evaluation environment with Go-Ethereum. First, we observed that a complex configuration is needed to set up and run benchmarks for different workloads. At the same time, the guidelines provided in the original package are insufficient. Secondly, BLOCKBENCH does not come with a default debugger as well as no debugging instructions are provided with the support.

We developed Debug-Bench to facilitate BLOCKBENCH with an instruction-based debug support in a separate layer.

Debug-Bench is a VS code (Visual Studio Code) extension that resides on top of the BLOCKBENCH framework as an independent layer, explained in Fig. 9. It communicates with the lower layers to provide a user-friendly instructional setup and debug support with Delve [90] (a debugger for Go programs) and GDB (GNU Debugger) [91]. This extension on the run-time connects to the BLOCKBENCH and the debugger to pass on the user-defined inputs and configurations. Additionally, Debug-Bench takes the Go `pprof` package as the foundation and provides input and configuration support to profile Go programs. This helps gain better insights into the benchmarked results to understand different modules and components of a program or application.

In order to debug Go-Ethereum, Debug-Bench requires installing Go and setting Gopath to the project directories first. Install Delve and then make and build Geth with the following commands.

```
$ make clean
$ go build -o./build/bin/geth -gcflags=
    all='-N -l' -v ./cmd/geth
```

This will build Geth and create a data file in the path. Now we need to initiate the Genesis block by starting Ethereum with the follwing command.

```
$ ./start_ethereum.sh
```

Once Genesis is initiated, we need to start Geth with Delve debugger with all the necessary parameters and configurations. A simple example of Geth start and node attachment with delve debugger is given in the below command.

```
$ dlv debug --headless --api-version=2
    --listen=:2345 --log -- --datadir=/
    home/user/.ethereum/data --nodiscover
     --rpcapi="db,eth,net,web3,personal,
    web3" --verbosity "5" --pprof --
    pprofaddr="127.0.0.1" --rpc --rpcaddr
     "localhost" --rpcport "8545" --
    rpccorsdomain "*" --gasprice 0 --
    maxpeers 32 --networkid 9119 --unlock
     "xxxxx" --password <(echo -n "") --
    mine --miner.threads 1
```

Where `--pprofaddr` is the IP address of the location where PPROF needs to be created. This file is used for profiling the execution results with the GO profiling tool. Furthermore, `"xxxxx"` is the address of the Ethereum account because, in this example, we have connected a node to the live Ethereum network in real-time.

We tested Debug-Bench with Ethereum blockchain for both Delve and GDB debuggers. Our extension is easy to use and works with minimal inputs. It facilitates the implementation of BLOCKBENCH with two types of debuggers. However, we are working to include more debuggers and evaluation frameworks in our future work.

VIII. CONCLUSION

This paper comprehensively reviews blockchain technology, explicitly highlighting resource-constrained devices' performance, data management, and storage concerns. We analyzed data processing and performance of the Ethereum blockchain on two separate devices having different computing power. Our representation is novel in providing a component-wise deep understanding of the design principles, which consequently aids in choosing the most suitable and efficient approaches to enrich low-power devices with blockchain. We measured the performance of the Ethereum blockchain in terms of CPU usage, latency, and execution time for both high-power and resource-constrained devices. We explained why current blockchain solutions are less fit to be adopted in resource-constrained devices, e.g., IoT devices and devices equipped with low-power processors (e.g., ARM processors). We developed Debug-Bench, the first VS Code extension that facilitates BLOCKBENCH in adding basic configuration to set up an evaluation environment for evaluating, debugging, and profiling blockchain applications. In the end, we list some possible research directions from our experience that could assist in filling the performance gaps between current and futuristic blockchain implementations.

REFERENCES

[1] C. Sillaber and B. Waltl, "Life cycle of smart contracts in blockchain ecosystems," *Datenschutz Datensicherheit*, vol. 41, no. 8, pp. 497–500, Aug. 2017.
[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Decentralized Bus. Rev., Bitcoin, Saint Kitts and Nevis, Tech. Rep. 21260, 2008.
[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.
[4] Ethereum. *Ethereum Blockchain App Platform*. Accessed: 2013. [Online]. Available: https://www.ethereum.org
[5] Hyperledger. (2016). *Blockchain Technologies for Business*. [Online]. Available: https://www.hyperledger.org
[6] Quorum. (2016). *Advancing Blockchain Technology*. [Online]. Available: https://www.jpmorgan.com/global/Quorum
[7] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. CCS*, 2016, pp. 3–16.
[8] B. D. Cahya Putri and R. Fitri Sari, "The effect of latency on selfish-miner attack on block receive time bitcoin network using NS3," in *Proc. 12th Int. Conf. Telecommun. Syst., Services, Appl. (TSSA)*, Oct. 2018, pp. 1–5.
[9] Facebook. (2012). *A Persistent Key-Value Store for Fast Storage Environments*. [Online]. Available: https://rocksdb.org
[10] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, "Blockchain and IoT integration: A systematic survey," *Sensors*, vol. 18, no. 8, p. 2575, 2018.
[11] C. Xu, K. Wang, P. Li, S. Guo, J. Luo, B. Ye, and M. Guo, "Making big data open in edges: A resource-efficient blockchain-based approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 870–882, Apr. 2019.
[12] 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*, May 2017, pp. 1085–1100.
[13] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. Annu. Tech. Conf.*, 2014, pp. 305–319.
[14] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, 1999, pp. 173–186.
[15] F. P. Junqueira, B. C. Reed, and M. Serafini, "ZAB: high-performance broadcast for primary-backup systems," in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2011, pp. 245–256.
[16] L. Lamport, "Paxos made simple, fast, and Byzantine," in *Proc. OPODIS*, vol. 3, 2002, pp. 7–9.

[17] E. Androulaki, A. Barger, V. Bortnikov, and C. Cachin, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, p. 30.
[18] G. O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, and S. Čapkun, "Misbehavior in bitcoin: A study of double-spending and accountability," in *Proc. TISSEC*, 2015, vol. 18, no. 1, p. 2.
[19] Paritytech. *Proof of Authority*. Accessed: 2017. [Online]. Available: https://github.com/paritytech/parity/wiki/Proof-of-Authority-Chains
[20] Hyperledger. (2015). *Poet: Proof of Elapsed Time*. [Online]. Available: https://intelledger.github.io/introduction.html#proof-of-elapsed-time-poet
[21] Ripple. (2013). *Ripple Consensus*. [Online]. Available: https://www.ripple.com
[22] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proc. 25th Secur. Symp.*, 2016, pp. 279–296.
[23] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-Ng: A scalable blockchain protocol," in *Proc. 13th Symp. Networked Syst. Design Implement.*, 2016, pp. 45–59.
[24] Tendermint. (2014). *Tendermint Consensus*. [Online]. Available: https://www.tendermint.com
[25] J. Behl, T. Distler, and R. Kapitza, "Scalable BFT for multi-cores: Actor-based decomposition and consensus-oriented parallelization," in *Proc. 10th Workshop Hot Topics Syst. Dependability (HotDep)*, 2014.
[26] W. Zhao, "Optimistic byzantine fault tolerance," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 31, no. 3, pp. 254–267, May 2016.
[27] M. Zbierski, "Parallel byzantine fault tolerance," in *Soft Computing Computer Information Science*. Cham, Switzerland: Springer, 2015, pp. 321–333.
[28] D. Larimer. (2017). *DPOS Consensus Algorithm*. [Online]. Available: https://how.bitshares.works/en/master/technology/dpos.html
[29] Peercoin. (2012). *Pioneer of Proof of Stake*. [Online]. Available: https://peercoin.net
[30] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *CoRR*, vol. abs/1710.09437, pp. 1–5, Dec. 2017.
[31] Jelurida. (2013). *NXT—The Blockchain Application Platform*. [Online]. Available: https://nxtplatform.org
[32] Parity. (2016). *Blockchain Infrastructure for the Decentralised Web*. [Online]. Available: https://www.parity.io
[33] Hyperledger. (2017). *Hyperledger Sawtooth*. [Online]. Available: https://github.com/hyperledger/sawtooth-core
[34] SGX. *Fintech—Blockchain Based Disruptive Technology*. Accessed: 2021. [Online]. Available: https://www2.sgx.com/research-education/sector
[35] NEM. (2015). *The Smart Asset Blockchain*. [Online]. Available: https://nem.io, 2015.
[36] Lisk. (2016). *Access the Power of Blockchain*. [Online]. Available: https://lisk.io
[37] Block. *Eosio Blockchain Protocol*. Accessed: 2016. [Online]. Available: https://eos.io
[38] Steemit. (2016). *Steem—Powering Communities and Opportunities*. [Online]. Available: https://steem.com
[39] BITSHARES. *Bitshares Blockchain: Industrial-Grade Decentralized Platform*. Accessed: 2014. [Online]. Available: https://bitshares.org
[40] ARK. *All-in-One Blockchain Solutions*. [Online]. Available: https://ark.io
[41] STELLAR. (2014). *Develop the World's New Financial System*. [Online]. Available: https://www.stellar.org
[42] CASPERLABS. (2014). *CBC Casper Proof-of-Stake*. Accessed: 2017. [Online]. Available: https://casperlabs.io
[43] Monax. *Eris-db—Permissioned Blockchain*. Accessed: 2016. [Online]. Available: https://github.com/christrewin/eris-db
[44] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Advances in Cryptology*. Cham, Switzerland: Springer, 2017, pp. 357–388.
[45] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2018, pp. 66–98.
[46] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability," in *Proc. 2018 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 913–930.
[47] A. Kolluri, I. Nikolic, I. Sergey, A. Hobor, and P. Saxena, "Exploiting the laws of order in smart contracts," in *Proc. 28th ACM SIGSOFT ISSTA*, 2019, pp. 363–373.
[48] M. Mulders. (2018). *Comparing Erc20/Erc223 and the New Ethereum Erc777 Token Standard*. [Online]. Available: https://www.cointelligence.com/content/author/michiel-mulders
[49] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Proc. DCCL*, vol. 310, 2016, pp. 1–5.
[50] SSC. (2015). *Stellar Smart Contracts*. [Online]. Available: https://www.stellar.org/developers/guides/walkthroughs/stellar-smart-contracts.html
[51] Wiki. *Turing Completeness*. Accessed: 2020. [Online]. Available: https://en.wikipedia.org/wiki/Turing_completeness
[52] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A survey of lightweight-cryptography implementations," *IEEE Design Test Comput.*, vol. 24, no. 6, pp. 522–533, Dec. 2007.
[53] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, 2001.
[54] H. Dobbertin, A. Bosselaers, and B. Preneel, "Ripemd-160: A strengthened version of ripemd," in *Proc. Int. Workshop Fast Softw. Encryption*, pp. 71–82, 1996.
[55] CoinSciences. (2015). *Open Platform for Building Blockchains*. [Online]. Available: https://www.multichain.com
[56] C. Prism. (2015). *Blockchain Technology for the Enterprise*. [Online]. Available: https://www.openchain.org
[57] Apache. *Kafka*. Accessed: 2012. [Online]. Available: https://kafka.apache.org.
[58] J. Göbel and A. E. Krzesinski, "Increased block size and bitcoin blockchain dynamics," in *Proc. 27th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, 2017, pp. 1–6.
[59] I. A. Seres, A. Dániel Nagy, C. Buckland, and P. Burcsi, "MixEth: Efficient, trustless coin mixing service for ethereum," in *Proc. Int. Conf. Blockchain Econ., Secur. Protocols*, p. 13:1–13:20, 2019.
[60] I. Bashir, *Mastering Blockchain*. London, U.K.: Packt, 2017.
[61] E. A. Brewer, "Towards robust distributed systems," in *Proc. 19th ACM Symp.*, vol. 7, 2000, pp. 343477–343502.
[62] P. Bailis and A. Ghodsi, "Eventual consistency today: Limitations, extensions, and beyond," *Queue*, vol. 11, no. 3, p. 20, 2013.
[63] *Mongodb*. Accessed: 2009. [Online]. Available: https://www.mongodb.com.
[64] CNCF. *Rethinkdb*. Accessed: 2009. [Online]. Available: https://www.rethinkdb.com
[65] M. Dai, S. Zhang, H. Wang, and S. Jin, "A low storage room requirement framework for distributed ledger in blockchain," *IEEE Access*, vol. 6, pp. 22970–22975, 2018.
[66] S. Wang, T. T. A. Dinh, Q. Lin, Z. Xie, M. Zhang, Q. Cai, G. Chen, B. C. Ooi, and P. Ruan, "Forkbase: An efficient storage engine for blockchain and forkable applications," *Proc. VLDB Endowment*, vol. 11, no. 10, pp. 1137–1150, 2018.
[67] HydraChain. *Permissioned Distributed Ledger Based on Ethereum*. Accessed: 2015. [Online]. Available: https://github.com/HydraChain/hydrachain
[68] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for iot," in *Proc. ACM IoTDI*, 2017, pp. 173–178.
[69] M. Conoscenti, A. Vetro, and J. C. D. Martin, "Blockchain for the Internet of Things: A systematic literature review," in *Proc. 13th AICCSA*, 2016, pp. 1–6.
[70] C. Roulin, A. Dorri, R. Jurdak, and S. Kanhere, "On the activity privacy of blockchain for IoT," *CoRR*, vol. abs/1812.08970, pp. 1–8, Dec. 2018.
[71] J. Kreku, V. A. Vallivaara, K. Halunen, J. Suomalainen, M. Ramachandran, V. Kantere, G. Wills, and R. Walters, "Evaluating the efficiency of blockchains in IoT with simulations," in *Proc. IoTBDS*, 2017, pp. 216–223, 2017.
[72] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of Things (IoT) security: Current status, challenges and prospective measures," in *Proc. 10th Int. Conf. for Internet Technol. Secured Trans. (ICITST)*, 2015, pp. 336–341.
[73] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *Proc. PerCom Workshops*, 2017, pp. 618–623.
[74] J. R. Douceur, "The sybil attack," in *Int. Workshop Peer Syst.*, 2002, pp. 251–260.
[75] S. Barber, X. Boyen, E. Shi, and E. Uzun, "Bitter to better—How to make bitcoin a better currency," in *Proc. Int. Conf.*, 2012, pp. 399–414, 2012.
[76] H. Spenkelinkl, D. D. Vries, and M. Berghuijs. *Blockchain Maturity Model*. Accessed: 2017. [Online]. Available: https://assets.kpmg/content/dam/kpmg/nl/pdf/2017/advisory/blockchain-maturity-model.pdf
[77] C. P. Team, "CMMI for systems engineering/software engineering/integrated product and process development/supplier sourcing," Softw. Eng. Inst., Chennai, India, Tech. Rep. CMU/SEI-2002-TR-012, 2002.

[78] T. Hardjono, A. Lipton, and A. Pentland, "Towards a design philosophy for interoperable blockchain systems," *CoRR*, vol. abs/1805.05934, pp. 1–5, Oct. 2018.
[79] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the Internet of Things: Perspectives and challenges," *Wireless Netw.*, vol. 20, no. 8, pp. 2481–2501, 2014.
[80] M. Imran and A. Mahmood, "An energy efficient model for WSN's monitoring applications," *IJCSIS*, vol. 14, no. 11, p. 363, 2016.
[81] IOTA. (2015). *The Next Generation of Distributed Ledger Technology*. [Online]. Available: https://www.iota.org
[82] Google. *Leveldb*. Accessed: 2014. [Online]. Available: https://github.com/google/leveldb
[83] Apache. *Couchdb*. Accessed: 2005. [Online]. Available: https://couchdb.apache.org
[84] Apache. *Spark*. Accessed: 2014. [Online]. Available: https://spark.apache.org
[85] Apache. *Hadoop*. Accessed: 2006. [Online]. Available: https://hadoop.apache.org
[86] Apache. *Flink*. Accessed: 2011. [Online]. Available: https://flink.apache.org
[87] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, "Edge analytics in the Internet of Things," *Pervasive Comput.*, vol. 14, no. 2, pp. 24–31, 2015.
[88] P. Gazi, A. Kiayias, and D. Zindros, "Proof-of-Stake sidechains," in *Proc. IEEE Symp. Secur. Privacy*, May 2019, pp. 139–156.
[89] A. Kiayias and D. Zindros, "Proof-of-work sidechains," in *Proc. Financial Cryptography Data Secur.*, 2020, pp. 21–34.
[90] D. Parker. *Delve Debugger*. Accessed: 2014. [Online]. Available: https://github.com/go-delve/delve
[91] R. Stallman. (1986). *Gnu Debugger*. [Online]. Available: https://www.gnu.org/software/gdb

MUHAMMAD IMRAN received the master's degree in computing from the Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, in 2016. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China. He holds two distinctions in his academic career. His research interests include distributed computing, blockchain for low-power computing, data management of distributed & large databases, scalable data-driven learning, and deep learning for reliable analytics.

BIN YAO received the Ph.D. degree in computer science from the Department of Computer Science, Florida State University, in 2011. He is currently a Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include management and indexing of large databases, query processing in spatial and multimedia databases, string and keyword search, and scalable data analytics.

WAQAS ALI is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China. His research interests include web semantics, data management of distributed & large databases, query processing and indexing, and deep learning for reliable semantics.

ADNAN AKHUNZADA (Senior Member, IEEE) is currently a Professional Member of the ACM with 13 years of research and development experience both in ICT industry and academia. He has a proven track record of high impact published research (i.e., U.S. patents, journals, transactions, reputable magazines, book chapters, conferences, and conference proceedings) and commercial products. His experience as an Educator and a Researcher is diverse that includes work as a Lecturer, a Senior Lecturer, a Year Tutor, an Occasional Lecturer at other engineering departments, as an Assistant Professor at COMSATS University Islamabad (CUI), a Senior Researcher at RISE SICS Västerås AB, Sweden, as a Research Fellow and the Scientific Lead at DTU Compute, Technical University of Denmark (DTU), as the Course Director of Ethical Hacking at The Knowledge Hub Universities (TKH), Coventry University, U.K., and a Visiting Professor having mentorship of graduate students, and supervision of academic and research and development projects both at UG and PG levels. He is also as an Associate Professor with the Faculty of Computing and Informatics, Universiti Malaysia Sabah, Malaysia. He has also been involved in international accreditation, such as an Accreditation Board for Engineering and Technology (ABET), and curriculum development according to the guidelines of ACM/IEEE. He is a PI of national and a Co-PI of several Swedish and Horizon 2020 EU funded projects. His main research interests include cyber security, secure future internet, artificial intelligence (i.e., machine learning, deep learning, and reinforcement learning), large scale distributed systems (i.e., edge, fog, cloud, and SDNs), the IoT, industry 4.0, and the Internet of Everything (IoE). He is also a member of technical program committee of varied reputable conferences, journals, and editorial boards.

MUHAMMAD KASHIF AZHAR is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China. He is also an Assistant Professor at the University of Education, D. G. Khan Campus. His current research interests include data management of distributed & large databases, scalable data-driven learning, and deep learning for reliable analytics.

MUHAMMAD JUNAID received the Ph.D. degree in computing from Iqra University, Islamabad, Pakistan. He is currently working as an Assistant Professor with the IT Department, The University of Haripur, Khyber Pakhtunkhwa, Pakistan. His research interests include cloud computing, blended learning, machine learning, swarm intelligence, general management, and information security.

UZAIR IQBAL received the Ph.D. degree in computer sciences (machine learning-based ECG analysis) from the University of Malaya, Malaysia, in 2020. From 2016 to 2021, he worked as a Lecturer with the Software Engineering Department, National University of Modern Languages, Islamabad, Pakistan. He is currently working as an Assistant Professor with the Faculty of Computer Sciences, National University of Computer and Emerging Sciences (NUCES) Chiniot–Faisalabad Campus, Pakistan. He analyzed ECG data streams to identify the correlation between multiple heart diseases using model-driven deep deterministic learning. He published a number of ISI Index research papers in top venue journals. Moreover, he is an Active Member of Research Community of the Institute for Systems and Technologies of Information, Control and Communication (INSTICC). Additionally, he is an Active Member of the European Society Cardiology (ESC).