POLITECNICO DI TORINO
Repository ISTITUZIONALE

Comparative analysis of permissioned blockchain frameworks for industrial applications

Original
Comparative analysis of permissioned blockchain frameworks for industrial applications / Capocasale, Vittorio; Gotta, Danilo; Perboli, Guido. - In: BLOCKCHAIN: RESEARCH AND APPLICATIONS. - ISSN 2096-7209. - STAMPA. - 4:1(2023), pp. 1-13. [10.1016/j.bcra.2022.100113]

Availability:
This version is available at: 11583/2973439 since: 2023-07-21T08:51:47Z

Publisher:
Elsevier

Published
DOI:10.1016/j.bcra.2022.100113

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

28 April 2024

Blockchain: Research and Applications 4 (2023) 100113

Contents lists available at ScienceDirect

Blockchain: Research and Applications

journal homepage: www.journals.elsevier.com/blockchain-research-and-applications

Research Article

Comparative analysis of permissioned blockchain frameworks for industrial applications

Vittorio Capocasale [a,*,1], Danilo Gotta [b,*,1], Guido Perboli [c,*,1]

[a] Department of Control and Computer Engineering, Politecnico di Torino, 10129 Turin, Italy
[b] Service Innovation, TIM, Turin, Italy
[c] Department of Management and Production Engineering, Politecnico di Torino, 10129 Turin, Italy

ARTICLE INFO

Keywords:
Blockchain performance evaluation
Hyperledger Besu
Hyperledger Fabric
Hyperledger Sawtooth
Quorum

ABSTRACT

Blockchain is a technology that creates trust among non-trusting parties without relying on any intermediaries. Consequently, it has attracted the interest of companies operating in a multitude of sectors. However, due to the number of different blockchain solutions that have emerged in the last few years and their rapid changes, it is challenging for such companies to orient their technological decisions. This paper presents a comparative analysis of the key dimensions—namely, governance, maturity, support, latency, privacy, interoperability, flexibility, efficiency, resiliency, and scalability—of some of the most-used permissioned blockchain platforms. Moreover, we present the results of a performance evaluation considering the following frameworks: Hyperledger Fabric 2.2, Hyperledger Sawtooth 1.2, and ConsenSys Quorum 21.1 (with both the GoQuorum client and the Hyperledger Besu client). The platforms were tested under similar conditions, and official releases were used, such that our findings provide a reference for companies establishing their technological orientation.

1. Introduction

At present, companies are undergoing radical transformations based on information sharing and digitalization. This is known as the Industry 4.0 revolution. The affordability of Internet of Things (IoT) and storage devices has allowed companies to gather enormous quantities of data. Such data can then be used to improve and optimize existing business processes, with huge cost savings. Consequently, data trustworthiness is fundamental and can be guaranteed by blockchain.

Blockchain is an evolving technology that allows for the creation of trust among non-trusting parties without relying on any intermediaries [1]. A blockchain can be described as a shared and distributed database: each non-trusting party can store and retrieve data from the database without worrying about tampering attempts [2].

Blockchain systems can be either permissionless or permissioned. In the former case, anyone can join the system and fully interact with the database, while in the latter case, it is possible to set up roles and policies to limit interactions with the database [3]. Permissioned blockchain frameworks are particularly relevant for the industry: often, companies need to share data among themselves while limiting or preventing external access to such data (e.g., for regulatory reasons).

At present, the blockchain landscape is quickly evolving: many blockchain-based solutions are available on the market, with new stable releases published every few months. Consequently, it is difficult for companies to orient their technological decisions, as keeping track of frequent updates and evaluating their practical impact is challenging. However, companies need a constantly updated overview of the various blockchain solutions in order to establish which ones fit a given use case. In particular, efficiency is a key factor that considerably limits which applications can exploit blockchain technology. Consequently, a procedure that allows for a fair evaluation of the performances of the various blockchain solutions is fundamental. These requirements have emerged from a collaboration between Politecnico di Torino and TIM S.p.A., one of the biggest telecommunication companies in Europe.

In particular, in industrial IoT applications, a specific blockchain solution must be carefully chosen. For example, in logistics, many transactions must be processed in a given time unit [2,4,5]. Contrary to what occurs in financial applications [6], blockchain must be integrated with

* Corresponding authors.
E-mail addresses: vittorio.capocasale@polito.it (V. Capocasale), danilo.gotta@telecomitalia.it (D. Gotta), danilo.gotta@telecomitalia.it (G. Perboli).

[1] Equally contributing authors.

https://doi.org/10.1016/j.bcra.2022.100113
Received 23 February 2022; Received in revised form 22 October 2022; Accepted 26 October 2022
2096-7209/© 2022 The Authors. Published by Elsevier B.V. on behalf of Zhejiang University Press. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

other technologies: IoT devices are necessary to collect data from physical assets [7,8], while artificial intelligence, analytics, and granular computing techniques can extract useful information from the collected data [9,10].

Nonetheless, comparative analyses of multiple blockchain frameworks are generally lacking. Due to the rapid nature of technical improvements, many comparative analyses and performance evaluations in the literature are already outdated. Moreover, many articles describe performance evaluations carried out on a single framework. This prevents a fair comparison among different frameworks, as the different authors have generally used different configurations and testing methodologies.

In this study, we fill the aforementioned gaps by providing an updated comparative analysis and a fair performance evaluation of various permissioned blockchain frameworks. The main contributions of this paper are as follows.

- A comparative analysis is presented, considering some of the most-used blockchain frameworks, namely, Hyperledger Fabric [11], Hyperledger Sawtooth [12], and ConsenSys Quorum (with both the GoQuorum client and the Hyperledger Besu client) [13]. The analysis assessed the following aspects: governance, maturity, support, latency, privacy, interoperability, flexibility, efficiency, resiliency, and scalability.
- A methodology is presented for performing a comparative performance evaluation of different blockchain frameworks. To the best of our knowledge, this methodology is the first to focus on the cross-framework fairness and comparability of the tests. In particular, this methodology is innovative as it allows for minimizing the differences among the different frameworks.
- We present one of the most comprehensive cross-framework performance evaluations in the literature. To fill the gaps in the literature, we tested recent releases of the frameworks. Moreover, to minimize the differences among the various frameworks, similar transactions were submitted and the same underlying hardware was used. Different blockchain nodes were deployed over the same industrial cloud infrastructure (Amazon AWS).

Thus, our findings offer a comprehensive overview of the analyzed frameworks, and this paper can be used by companies as a guide for their technological choices.

The remainder of this paper is structured as follows: Section 2 introduces the key concepts related to blockchain technology. Section 3 discusses the relevant literature and the related gaps. Section 4 presents the comparative analysis, and Section 5 describes the performance evaluation of the various frameworks. Finally, Section 6 presents our conclusions and future developments.

2. Background

In this section, we briefly review the main concepts related to blockchain, consensus algorithms, smart contracts, and performance metrics. For reader convenience, this section also includes an introduction to the frameworks analyzed in the context of this work.

2.1. Blockchain

Blockchain belongs to the distributed ledger technology (DLT) class [14]. A DLT is a distributed database that is structured as a ledger (see Fig. 1). This means that it records the whole history of modifications (also called transactions) to the data it stores, and multiple copies of the ledger are available. Each copy is managed by an entity called a peer.

Blockchain groups transactions into blocks, which are then added to the ledger, one after the other [16]. As each block contains the hash of its predecessor, each block cannot be altered without also altering all the subsequent ones. It is easy to alter a block and all its successors when all

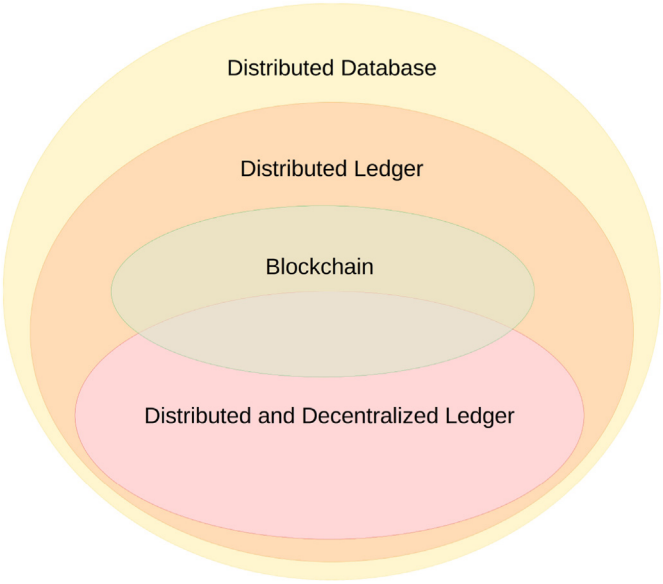

Fig. 1. The relationships among the distributed database, distributed ledger technology (DLT), and blockchain technologies. In particular, a DLT is a distributed database structured as a ledger. A DLT can be decentralized or not, depending on its governance model. A blockchain is a DLT that uses a list of blocks to represent the ledger. Each block contains the hash of the header of its predecessor. Blockchain technology is associated with many interesting properties, such as immutability; however, such properties only characterize decentralized blockchains [15].

the copies of the ledger are managed by a single entity or by trusting parties. However, altering a block and all its successors can be nearly impossible when the ledger is managed by multiple non-trusting parties. For this reason, the properties characterizing a blockchain system change drastically depending on the governance model in use. This is better explained in Section 2.2.

A blockchain is usually composed of two databases: the history database, which is the actual ledger, and the state database, which holds the current values of the data stored in the history database. It is possible to create blockchain systems without a state database, but this approach has significant performance drawbacks. The state database acts as a cache that allows for fetching the latest values of the data without needing to read the whole ledger. A blockchain address identifies a specific portion of the data contained in the state database.

Blockchain frameworks rely on the following key components [16]:

- digital signatures, which allow for authentication of the transactions;
- cryptographic hash functions, which allow for the creation of the append-only ledger structure;
- consensus algorithms, which are used to decide the order of transactions to process.

2.2. Blockchain governance

Governance describes the power to control, coordinate and direct a blockchain system [17]. According to its governance model, a blockchain system can be [18,19]:

- **public**—Any peer can join the blockchain system and participate in the consensus protocol. A public blockchain can be used to solve trust issues among its participants, as any interested party can join the network, obtain a full copy of the ledger, and autonomously validate the transactions it contains;
- **consortium**—The blockchain system is managed by some well-identified peers who can set the rules for interacting with the

ledger and for participating in the consensus. A consortium blockchain can be used to solve trust issues among the consortium members, but parties that are external to the consortium still need to trust the consortium;
- **private**—The blockchain system is managed by a single party. Consequently, the system is centralized from a governance standpoint and requires all the participants to trust the managing party.

As public blockchains do not restrict access to their ledgers, they are permissionless blockchains. Similarly, private and consortium blockchains are permissioned blockchains, as they allow access control mechanisms to be set up. However, private and consortium blockchains are rarely interchangeable technological solutions and should be used in different cases.

2.3. Blockchain properties

The following properties make blockchain particularly interesting in industrial fields [16,19,20]:

- **redundancy** and **persistency**—Each peer keeps a copy of the ledger, which reduces the risk of data losses;
- **decentralization**—Each peer has control over a single copy of the ledger, not the ledger itself. Notably, this property is a consequence of the decentralized governance model of a blockchain system and not of its distributed nature. Consequently, private blockchains are not decentralized;
- **authenticity**—Transactions are digitally signed;
- **autonomy**—Peers can submit transactions without relying on trusted third parties;
- **immutability**—Data can only be added to the ledger, but not modified, as the hash of an altered block would not match the one stored in its successor. However, rewriting the whole chain of hashes is possible: in a private blockchain, the managing party may do so individually; in a consortium blockchain, the consortium should collude in order to do so; and, in a public blockchain, the majority should collude in order to do so (51% attack);
- **transparency** and **auditability**—Each peer has direct access to its own copy of the ledger. Moreover, it is possible to know the status of the ledger at any given point in the past, as the whole history of modifications is recorded;
- **resiliency**—To counterfeit the ledger, it would be necessary to coherently modify the majority of its copies. As observed for the immutability property, the resiliency of a blockchain system is proportional to its decentralization;
- **standardization**—As many peers must keep identical copies, they must all agree on the encoding of data.

2.4. Consensus algorithms

As a blockchain system is managed by many peers, such peers need to find an agreement on the order of transactions to process. The decision is made through the use of a consensus algorithm. The decision can have [21,22]:

- **deterministic finality**—Once made, the decision is irreversible;
- **probabilistic finality**—Once made, the probability of reverting the decision decreases over time.

Moreover, the peers should be able to make a common decision, even if some of them do not participate in the consensus protocol or try to disrupt it. Consequently, consensus algorithms can be [21,22]:

- **Crash Fault Tolerance (CFT)**—The consensus can tolerate the crash of some peers. Raft [23] and proof-of-elapsed-time CFT (PoET CFT) [24] are examples of CFT algorithms.
- **Byzantine Fault Tolerance (BFT)**—The consensus can tolerate the crash of some peers or their malicious behavior. Ethash [25], Clique [26], Practical Byzantine Fault Tolerance (PBFT) [27], Istanbul Byzantine Fault Tolerance (IBFT) [28], and proof-of-elapsed-time SGX (PoET SGX) [24] are examples of BFT algorithms.

Many blockchain frameworks offer the possibility of choosing among CFT and BFT consensus algorithms; however, it is not possible to assume the absence of malicious behaviors among non-trusting parties. Thus, only BFT algorithms should be used in a decentralized blockchain system.

2.5. Smart contracts

Smart contracts can be described as tamper-proof computer programs [29]. The smart contract concept was introduced before the blockchain concept [30]; however, as guaranteeing the tamper-proof property is difficult, smart contracts did not attract much interest at first. By coupling smart contracts and blockchains, it is possible to process data while guaranteeing their integrity and availability. Confidentiality can also be preserved, but it requires implementing additional cryptographic techniques that are rarely available by default. Among other applications, smart contracts could allow for the automation of legal contracts [31].

2.6. Performance metrics

Once submitted to a blockchain system, a transaction can be in one of the following states:

- pending—The transaction has not yet been added to a block;
- discarded—The transaction is invalid and has not altered the ledger;
- committed—The transaction is valid and has been added to a block;
- consolidated—The transaction is valid and is permanently stored in the blockchain. In the case of deterministic finality, a transaction is consolidated as soon as it is committed. In the case of probabilistic finality, a transaction is consolidated only after it is committed.

According to Ref. [32], the key metrics for blockchain systems are read latency, read throughput, transaction latency, and transaction throughput. In a blockchain system, transaction throughput is defined as the total amount of transactions consolidated in the time unit, and transaction latency is defined as the time needed by a transaction to become consolidated.

2.7. Blockchain frameworks

2.7.1. Hyperledger Fabric

Fabric [11] is an open-source framework designed to address common industrial needs, such as identity management, the definition of roles and policies, performance, and data confidentiality. Fabric belongs to the Hyperledger ecosystem, which is “an open-source community focused on developing a suite of stable frameworks, tools, and libraries for enterprise-grade blockchain deployments” [33]. Hyperledger Fabric offers a modular and scalable architecture. It supports smart contracts that can be written in a variety of widely adopted programming languages. To share data with only a subset of the nodes of the blockchain system, Fabric allows for the sending of private transactions (private data collections) or the creation of parallel and independent lightweight chains (channels). Fabric supports the following CFT consensus algorithms: Raft, Kafka (deprecated), and Solo (deprecated). A BFT consensus is planned for the future [34]. At the time of writing, version 2.3.2 was the latest available.

Fabric distinguishes between two types of nodes: orderers and peers. Peers are in charge of executing transactions and keeping a copy of the ledger, whereas orderers are in charge of creating blocks. Fabric processes transactions in three steps:

- execute—Each type of transaction is associated with an endorsement policy. The endorsement policy defines which peers must execute a given transaction. To submit a transaction, a client has to send it only to the endorsing peers. This allows for the sacrifice of decentralization for scalability. The endorsing peers process the transaction without updating their copy of the ledger. Then, they send a signed message back to the client, which must be delivered to the orderers;
- order—The orderers create blocks by ordering the endorsed transactions received. Once created, a block is broadcast to all the peers of the channel;
- validate—Each peer checks the correctness of each transaction within the received block and updates its copy of the ledger. Among other checks, transactions that have a read or write conflict with a previous transaction of the same block are considered invalid.

2.7.2. Hyperledger Sawtooth

Sawtooth [12] is an open-source framework designed for flexibility and separation of concerns; it abstracts the application layer from the security layer. This allows for the easy creation of blockchain systems that rely on dynamically replaceable components. As with Fabric, Sawtooth is a Hyperledger framework. Sawtooth offers the possibility of writing smart contracts in a variety of programming languages. It also offers a parallel scheduler that can improve the performance of the framework. Sawtooth supports both BFT (PBFT and PoET SGX) and CFT (PoET CFT and Raft) consensus algorithms. The transaction processing strategy applied by Sawtooth is the standard order—execute—validate. Moreover, Sawtooth processes transactions in batches (i.e., groups of transactions that must all be completed together or not at all). At the time of writing, version 1.2.6 was the latest available.

The Sawtooth framework offers the following modules:

- validator component, which schedules transactions and manages the ledger;
- consensus engine, which implements the consensus algorithm;
- REST API component, which simplifies the interaction of the clients with the validator component;
- transaction processor (TP), which implements the smart contract logic.

2.7.3. ConsenSys Quorum

Quorum [13] is an open-source blockchain protocol based on the Ethereum protocol. It allows for the design of high-performance permissioned blockchain systems that provide support for data confidentiality. Quorum can also be used for interactions with the Ethereum network. Moreover, Ethereum smart contracts can be effortlessly migrated to Quorum. Quorum comprises two distinct blockchain projects: the first is based on GoQuorum [35], an Ethereum client originally developed by J.P. Morgan and currently maintained by ConsenSys (that renamed it from Quorum to GoQuorum), which is implemented in Go. The Tessera module can be used to send private transactions and keep data confidential. GoQuorum supports the following consensus algorithms: Raft (CFT), Clique (BFT), and IBFT version 1.0 (BFT). At the time of writing, version 21.4.2 was the latest available. The second project is based on Hyperledger Besu [36], an Ethereum client implemented in Java. As with Fabric and Sawtooth, Besu is a Hyperledger project. The Orion module can be used to send private transactions and to keep data confidential. Besu supports the following BFT consensus algorithms: Ethash, Clique, and IBFT (versions 1.0 and 2.0). At the time of writing, version 21.1.7 was the latest available.

In this paper, when both projects share a common feature, the generic word Quorum is used, while the words GoQuorum and Besu are used to refer to one of the two specific implementations. Quorum's transaction processing strategy is the standard order-execute-validate.

3. Related work

Permissioned blockchain frameworks have been increasingly receiving interest from various companies. However, such frameworks must process production workloads to replace existing solutions. Thus, many studies have addressed the topic of assessing the performance of existing blockchain frameworks.

For the sake of brevity, Table 1 summarizes the main performance evaluations available in the literature. For each paper, the table presents the analyzed framework(s). Moreover, the table shows which studies analyzed more than one framework, which performed an experimental performance evaluation, which used recent releases of the frameworks, and which ones described a methodology to minimize the differences among the different frameworks.

Performance evaluations of permissioned blockchain frameworks appeared in the literature early on. Pongnumkul et al. provided a performance evaluation of Fabric v0.6.0 and enterprise Ethereum (Geth v1.4.18) on a single blockchain node, which is not relevant for industrial use cases [43]. Dinh et al. introduced Blockbench, a tool for analyzing permissioned blockchain frameworks. The authors used their tool to evaluate the performance of the following blockchain frameworks: Fabric v0.6.0-preview, Geth v1.4.18, and Parity v1.6.0 [42]. Unfortunately, such performance evaluations are outdated, as suggested by the versions of the employed frameworks.

In many cases, authors tried to improve the official framework releases. Sousa et al. introduced a BFT algorithm for Fabric v1.0. The related performance evaluation, however, was focused on Fabric's ordering service [40]. Thakkar et al. presented an in-depth study on Hyperledger Fabric v1.0 and showed how the various configuration parameters affected the overall performance. The authors also suggested some improvements that were subsequently adopted in Fabric v1.1 [38]. Gorenflo et al. introduced FastFabric, an optimized version of Hyperledger Fabric 1.2, which allowed the authors to process almost 20,000 transactions per second. However, some of the proposed optimizations may raise concerns (e.g., keeping the state database in volatile memory) [39]. Kwon and Yu proposed some optimizations for the order and validated phases of Hyperledger Fabric v1.3. A performance evaluation on a network of four nodes and one Kafka orderer was used to show the benefits of the proposed optimizations [57]. However, stable and long-term supported releases are often preferred for industrial applications. Thus, performance evaluations performed on official releases are more appreciated.

Some authors focused on the analysis of the performance of a single framework instead of comparing multiple ones. Baliga et al. conducted an in-depth performance evaluation of Quorum (GoQuorum client v2.0) with both IBFT in a four-node network and Raft in a three-node network. The authors tested both private and public transactions and used four different workloads [41]. Similarly, Mazzoni et al. studied Quorum with all Raft, IBFT, and Clique consensus [6]. Mera conducted a performance evaluation of Quorum (GoQuorum client v2.2.1) with Raft consensus on a network of three nodes in three different settings (local nodes, cloud nodes, and virtual nodes) [37]. Wang and Chu evaluated Fabric v1.4.3. The authors tested all the consensus algorithms available in Fabric (Solo, Kafka, and Raft) and the impact of different endorsement policies [54]. Nakaike et al. introduced HLF-GLDB, a benchmark tool that allows simulating the database access patterns of Hyperledger Fabric. The authors used HLF-GLDB to discover some bottlenecks in the Fabric v1.4.4 platform [53]. Guggenberger et al. conducted an in-depth performance evaluation of Fabric 2.0. In their work, the authors examined the effect of various network sizes and underlying hardware, crashing nodes, network delays, private transactions, and varying workloads [56]. Shi et al. evaluated Sawtooth v1.1 with PoET CFT. The authors tested the transaction throughput under different conditions (network size, underlying

Table 1
Summary of the studies dealing with tests on blockchain frameworks.

Ref.	Framework	Multiple frameworks	Experimental performance evaluation	Recent releases	Cross-framework methodology
[15]	Sawtooth v1.0.5 with PoET CFT	No	Yes	No	No
[37]	GoQuorum v2.2.1	No	Yes	No	No
[38]	Fabric v1.0	No	Yes	No	No
[39]	Fabric v1.2	No	Yes	No	No
[40]	Fabric v1.0	No	Yes	No	No
[41]	GoQuorum v2.0	No	Yes	No	No
[42]	Fabric v0.6.0-preview, enterprise Ethereum (Geth) v1.4.18, Parity v1.6.0	Yes	Yes	No	No
[43]	Fabric v0.6.0 and enterprise Ethereum (Geth) v1.4.18	Yes	Yes	No	No
[44]	Fabric v0.6 and Fabric v1.0	No	Yes	No	No
[45]	Fabric v1.2	No	Yes	No	No
[46]	Fabric v1.4	No	Yes	No	No
[47]	Fabric v1.0	No	Yes	No	No
[48]	Sawtooth v1.0	No	Yes	No	No
[49]	GoQuorum v2.0.2	No	Yes	No	No
[50]	Sawtooth v1.0.5	No	Yes	No	No
[51]	Fabric, Sawtooth, Burrow, BigchainDB, MongoDB (Sep 2019)	Yes	Yes	No	No
[52]	Sawtooth v1.1.2, enterprise Ethereum (Geth) v1.8.21, enterprise EOS v1.5.3	Yes	Yes	No	No
[53]	Fabric v1.4.4	No	Yes	No	No
[54]	Fabric v1.4.3	No	Yes	No	No
[55]	Sawtooth v1.1 with PoET CFT	No	Yes	No	No
[56]	Fabric v2.0	No	Yes	Yes	No
[57]	Fabric v1.3	No	Yes	No	No
[58]	Fabric v1.4.4, Sawtooth v1.2, Indy v1.12.0, Parity v2.5.10, GoQuorum v2.3.0, enterprise Ethereum (Geth) v1.9.8	Yes	Yes	No	No
[59]	Fabric v2.2.2 and Sawtooth v1.2.3	Yes	Yes	Yes	No
This work	Fabric v2.2.2, Sawtooth v1.2.3, Besu v21.1, GoQuorum v21.1	Yes	Yes	Yes	Yes

Details about which studies analyzed more than one framework, which presented an experimental performance evaluation, which used recent releases of the frameworks, and which described a methodology to level the differences among the different frameworks are also provided. PoET: proof-of-elapsed-time, CFT: crash fault Tolerance.

hardware, network bandwidth, cloud service, and datacenter location) [55]. Such works provide meaningful insights into the configuration of a given framework but are less useful for comparing different frameworks.

Some authors evaluated the performance of multiple frameworks. Polge et al. presented a comparative analysis of Fabric, enterprise Ethereum, Quorum, MultiChain, and Corda in the following dimensions: community activity, adoption, performance, and privacy support. The authors, however, did not perform an experimental performance evaluation but conducted their analysis according to the results presented in other studies [60]. Monrat et al. performed a performance evaluation on the following frameworks: enterprise Ethereum, Corda, Fabric, and Quorum (GoQuorum client). The authors used the Microsoft Azure Platform for deploying networks of various sizes. However, except for Corda, they did not provide framework versions [61]. Benahmed et al. compared Sawtooth v1.1.2, enterprise Ethereum (Geth v1.8.21), and enterprise EOS (client v1.5.3). The authors described the usability, support, and documentation of the platforms they studied and tested their throughput, scalability, CPU, and memory usage [52]. Rasolroveicy and Fokaefs studied blockchain frameworks and MongoDB for IoT-based applications. The study focused on Fabric, Sawtooth, Burrow, and BigchainDB (Sep 2019) [51]. The results provided by such studies are not comparable, as different testing methodologies are employed.

In addition to Blockbench [42], other blockchain benchmark tools are available in the literature. A few of the previously discussed studies [6, 41,61] used Hypeledger Caliper [62], which provides a set of predefined workloads and is compatible with multiple frameworks. BCTmark [63] focuses on abstracting the underlying blockchain frameworks and improving benchmark portability and was used by its creators to perform a performance evaluation of enterprise Ethereum and Hyperledger Fabric. Nonetheless, the tool is in the experimental stage, lacks clearly defined performance metrics, and does not offer standard workloads [64]. The Distributed Ledger Performance Scan (DLPS) [58] allows setting up blockchain networks relying on different frameworks and submitting standard workloads to them. The DLPS defines clear metrics for the evaluation of blockchain systems and uses an adaptive testing strategy that matches the output throughput to the input one. The authors claim that such an approach allows for maximizing the performance of the tested frameworks: Hyperledger Fabric, Hyperledger Sawtooth, Hyperledger Indy, Quorum (GoQuorum client), and Ethereum (Geth and Parity clients). Ref. [64] summarizes the main benchmark tools for permissioned blockchain frameworks.

While all such benchmark tools allow the generation of similar workloads on different frameworks, they do not provide insights into how to set up blockchain networks to obtain similar degrees of security, distribution, and decentralization across different frameworks. Thus, the results obtained on different frameworks with such tools may not be comparable, as frameworks allow for trading security and decentralization for efficiency. This paper tries to overcome such a limitation by proposing a cross-framework methodology to level the differences among different frameworks, which enables meaningful comparisons among their performances.

We partnered with TIM, one of the biggest telecommunication companies in Europe. TIM has multiple business units working on blockchain-related topics. According to our partner, choosing the right blockchain framework can be challenging due to the lack of comparative analyses. Another international partner and other studies [65] sustained such a hypothesis. Moreover, our review of the literature highlights the following gaps:

- comparative analyses are scarce;
- some analyses focus on very specific applications;
- often, tweaked versions of the frameworks are tested. Therefore, the results of such tests are not particularly useful when only official and supported releases of the frameworks are used;
- the frameworks are tested using different methods and under different conditions, preventing any comparison (even qualitative);

- some analyses are outdated and are no longer meaningful.

Nonetheless, the problem of assessing the performance of the various blockchain frameworks is important, as demonstrated by the large number of articles addressing this issue. We respond to these needs by presenting a general methodology for evaluating blockchain frameworks in industrial use cases. We used this methodology to conduct a performance evaluation of some of the most commonly industrially adopted blockchain frameworks.

4. Comparative analysis

This section presents a comparison of the blockchain frameworks considered in this study from a functional and high-level point of view. We underline that such an analysis is meant to highlight similarities and differences across the frameworks, not to elect winners.

4.1. Governance

As discussed in Section 2.2, governance describes the power to control, coordinate, and direct a blockchain system [17]. In blockchain systems, decisions are made by majority voting, and consensus algorithms are voting mechanisms [66]. Thus, analyzing consensus algorithms is fundamental for understanding the governance model of a blockchain system. We listed the consensus algorithms offered by each framework in Section 2.7.

As it currently lacks an official implementation of a BFT consensus algorithm, Fabric must be considered a private blockchain, even if the execution and validation steps can be fully decentralized. “Hyperledger Fabric is not reliable in an environment where an ordering service may be hacked” [67].

Sawtooth and Quorum, if deployed with a BFT consensus, can be used to build both public and consortium blockchain systems. Consequently, they can be used by non-trusting parties to resolve their trust issues.

4.2. Maturity

Maturity identifies the production readiness of blockchain frameworks. Fabric, Sawtooth, and Quorum are all production ready, according to their documentation and version numbers [34,35,68]. Fabric is probably the most widespread and used technology among the three, as proved by its number of implemented use cases [69,70]. Quorum is also used commonly in industry [13,69]. Sawtooth is somewhat less adopted in the industry [69,70] but has been widely adopted in the academic world [15,71–73].

4.3. Support

In this study, support describes to what extent blockchain frameworks streamline adoption in terms of both technological improvements and user experience.

Fabric, Quorum, and Sawtooth are all active projects, and they are supported by both official and unofficial channels.

In our opinion, the Fabric framework is well documented. However, setting up a system from scratch may be a non-trivial task. The official documentation on this matter can possibly be improved.

In our opinion, the Sawtooth framework is well documented and easy to set up. The documentation provides tutorials to set up a test system and describes the majority of the options needed to configure a custom production system in detail. Sawtooth nicely abstracts and separates the various blockchain layers (e.g., networking, security, and smart contracts). Consequently, it is an excellent framework for understanding blockchain technology.

In our opinion, the Quorum framework is partially documented: it relies on the Ethereum documentation for many core concepts, whereas it focuses its documentation on its peculiarities (e.g., privacy features). Different from Fabric, it provides many tutorials for setting up a test system, clearly explaining each step of the process. Overall, the Quorum documentation is not as detailed as that of Fabric or Sawtooth and focuses on a more practical approach.

Concerning the community activity, a depiction is given by the GitHub developers' analysis reported in Refs. [74,75]. As witnessed by Fig. 2, Fabric and Besu are gaining support, GoQuorum is stable, and Sawtooth is experiencing a downtrend. As Sawtooth's data in 2020 are not present in the reports, we extracted them directly from GitHub and represented the approximate trend with a dashed line.

4.3. Latency

We defined latency in Section 2.6. We note that transaction finality can be probabilistic or deterministic, depending on the consensus algorithm used. Probabilistic finality improves scalability and offers a higher transaction throughput, but also has a higher transaction latency [76]. PBFT [27], IBFT [28], and Raft [23] have deterministic finality, whereas Clique [26], Ethash [25], and PoET (both CFT and SGX) [24] have probabilistic finality.

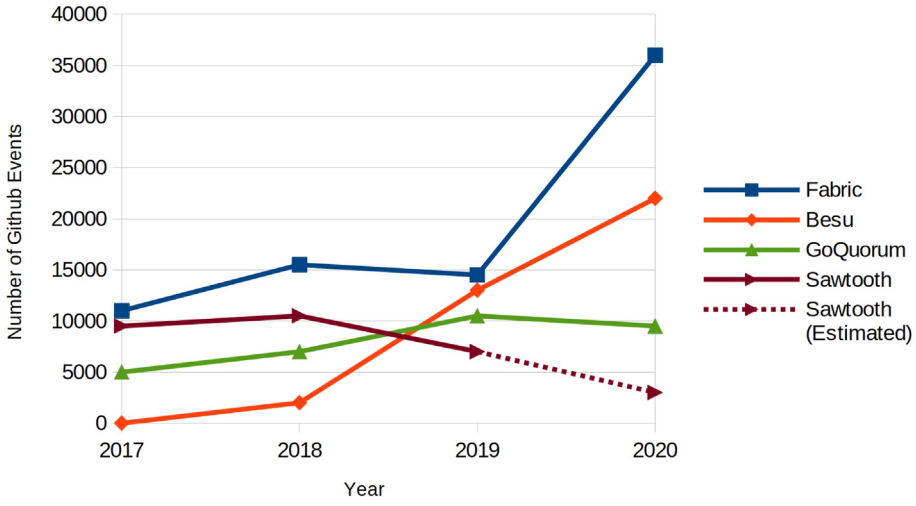

Fig. 2. Blockchain developer activity (e.g., commits, pull requests, forks, and so on) on GitHub in 2017–2020. The image shows which communities were the most active in the blockchain landscape. Sources: [74,75].

4.5. Privacy

In this paper, privacy refers to the possibility of sharing data with only a subset of the participants of a blockchain system. The main strategies applied to accomplish this goal are as follows [12]:

- share the hash of the data with all of the peers, and the actual data only with those of interest—This is the underlying strategy of Fabric's private data collections [77] and Quorum's Orion and Tessera modules [78];
- create a separate system—This is usually costly and may pose security concerns due to the reduced size of the system. To mitigate the cost drawback, Fabric offers the possibility of creating channels. Channels are separate blockchains, each with its own ledger; however, channels can reuse some common components. Consequently, multiple channels are less demanding in terms of hardware requirements, compared to separate blockchain systems [11];
- store ciphered data—This is always possible, but the encryption process must be handled by the client and cannot be managed by the framework. In Sawtooth, this is the only possible strategy [12].

4.6. Interoperability

Interoperability refers to the possibility of atomically transferring data across multiple blockchains [79]. Cross-chain communication is similar to interoperability but does not require atomicity [80]. The interested reader may find formal definitions of the two concepts in more technical studies [79,80]. The analyzed frameworks do not offer any features that simplify blockchain interoperability or cross-chain communication.

Interoperability can be partially achieved by leveraging cross-chain communication protocols or additional assumptions based on game theory and the trustworthiness of third parties. Protocols such as notary schemes or hash-locking [81] belong to such a category and have some major drawbacks. In particular, they are limited to some specific use cases, are very inefficient, introduce trusted third parties, or violate the atomicity of cross-chain transactions. Moreover, they do not allow the transfer of the history of an asset across different blockchain systems, which hinders their transparency and verifiability properties.

Full blockchain interoperability is impossible to achieve or requires merging the existing ledgers [79]. Similarly, cross-chain communication is impossible without relying on trusted third parties [80].

4.7. Flexibility

Flexibility refers to the possibility of replacing existing components or adding features to a blockchain framework.

Sawtooth is the most flexible, as it is composed of several components that can be dynamically replaced. Moreover, it allows for the specification of the settings both on- and off-chain. On-chain settings can be dynamically configured. Sawtooth allows for the specification of dependencies among transactions and submitting batches, which are groups of transactions that must be performed as a whole [12].

Quorum is flexible, as it allows for the configuration of many parameters, including the consensus algorithm. Moreover, both Quorum clients support the installation of plugins, which extend their set of functionalities [35,36].

Fabric is flexible, as it allows for the configuration of many parameters, including the consensus algorithm and the state database (LevelDB or CouchDB). Moreover, smart contracts are run as a separate component [11].

4.8. Efficiency

Efficiency indicates the quantity of information that a blockchain framework can process in the time unit and is analyzed in more detail in Section 5. For convenience, we report some observations here:

- the choice of the smart contract programming language has a relevant impact on the overall performance;
- Fabric and GoQuorum perform well in all tests;
- Besu performs well with light transactions but suffers a significant performance decay for heavier tasks;
- Sawtooth performs poorly, but much better results can be obtained by submitting larger batches [59]. Furthermore, due to the limited number of vCPUs used, Sawtooth's parallel scheduler has not been properly exploited.

4.9. Resiliency

Resiliency is the property of withstanding unexpected errors and malicious attacks. As discussed in Section 2.4, consensus algorithms can be CFT or BFT. At present, Fabric only offers official implementations of CFT algorithms, whereas Sawtooth and Quorum offer both possibilities.

It should be noted that elliptic-curve cryptography, which is the commonly accepted standard in blockchain systems and is used by all frameworks, is not quantum-safe [82].

4.10. Scalability

Scalability identifies the possibility of increasing the size of a blockchain network while minimizing the negative impacts on the other properties of the system (e.g., efficiency). According to the scalability trilemma, the scalability of a blockchain system can only be improved by sacrificing decentralization or security [2]. Using a CFT instead of a BFT algorithm is an example of a tradeoff between decentralization and scalability, which Fabric, Quorum, and Sawtooth all allow.

Fabric's endorsement policies can be considered another method for improving scalability: different nodes can be used to execute distinct sets of transactions in parallel. Additionally, channels can be used to improve scalability. This approach is equivalent to creating separate blockchain systems [11].

Quorum's IBFT consensus algorithm allows the set of nodes participating in the consensus protocol to be dynamically changed. This can be used to keep a small (and, thus, efficient) set of consensus nodes while giving all peers the possibility of being part of such a set for a limited amount of time [28].

As the set of consensus nodes in Sawtooth's PBFT is an on-chain setting, it can be dynamically updated, producing a behavior similar to that of Quorum's IBFT.

5. Performance analysis

As stated in Sections 1 and 3, a standard methodology to compare the performance of multiple blockchain frameworks was still missing, such that deciding which blockchain to use is difficult. Moreover, the limited interoperability among the different blockchain solutions makes this choice even more important.

This section describes the testing environment and the tests performed on the various frameworks. Notably, the tests relied on the same type of virtual machine, while the smart contract had to be implemented in every framework. Moreover, the configurations of the various frameworks were not tuned, as different frameworks offer different configuration settings, which modify the behavior of the system in different ways.

Concerning the frameworks used in the tests, we considered some of the most-used blockchain frameworks: Hyperledger Fabric, Hyperledger Sawtooth, and ConsenSys Quorum (with both the GoQuorum client and the Hyperledger Besu client).

5.1. Testing environment

To perform the tests, we constructed a network consisting of four AWS instances. The instances belonged to the same availability zone and to the same virtual private cloud (VPC). Each instance was an r5a.large virtual machine, with 2 vCPUs, 16 GB of RAM, and 50 GB SSD. The testing environment infrastructure is shown in Fig. 3.

The following settings describe the test environment used for the performance evaluation.

- Number of instances: 4.
- Network topology: complete graph, with instances hosted in the same availability zone.
- Instance type: AWS r5a.large.
- CPU (single instance): AMD EPYC 7000, 2.5 GHz, and 2 vCPUs.
- RAM (single instance): 16 GB.
- DISK (single instance): 50 GB gp2 SSD (EBS volume).
- OS: Ubuntu 20.04.2 LTS.
- Docker: 20.10.3, build 48d30b5.
- Docker-compose: version 1.28.4, build cabd5cfb.
- Node: v10.24.
- Go: go1.13.
- Java: openjdk v1.8.0_292.
- Solidity: 0.8.0+commit.c7dfd78e.Emscripten.clang.

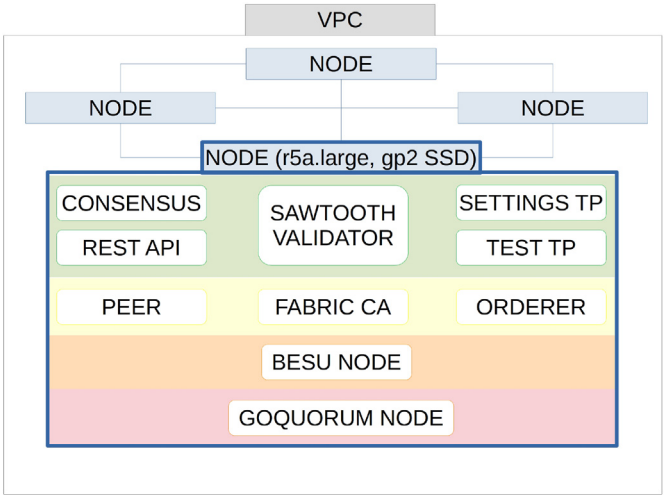

Fig. 3. The testing environment architecture used in this study. A network of four virtual machines was created on the AWS, where each virtual machine had multiple components. Components that belonged to the same framework are represented by the same color. The Sawtooth node was composed of a validator, a consensus engine, a REST API, and two transaction processors: one to manage on-chain settings and one to process the transactions of the tests. The Fabric node was composed of a peer, an orderer, and a certificate authority. Both the GoQuorum and Besu nodes consisted of a single component.

5.2. Methodology

As discussed in Section 4.10, blockchain frameworks allow sacrificing decentralization and security for efficiency and scalability. It is easy to prove that a given framework is more efficient than another when the former is configured to scale while the latter is configured to be secure and decentralized. Thus, comparing the performances of different frameworks is meaningless unless similar conditions are guaranteed across all the frameworks. Guidelines are available for the performance evaluations of a single framework [32], and some multi-framework benchmark tools have been implemented [42,58]. However, such guidelines and tools do not provide a methodology for setting up equivalent testing environments for the performance comparison of different blockchain frameworks. To fill this gap, we introduce a new methodology. We address the following concerns.

- Node functional requirements—Frameworks are composed of multiple modules, which must be assigned to hardware resources. However, some frameworks are more modular than others. Thus, it is necessary to define a blockchain node in terms of its functional requirements, which allows the creation of classes of equivalent modules across different frameworks. This allows for the assignment of modules to hardware resources following a consistent method across multiple frameworks. To the best of our knowledge, no other study has tackled this issue.
- Distribution requirements—It is necessary to use the same network topology and geographic distribution of nodes across multiple frameworks. We underline that enforcing the same geographic distribution is only possible after providing a cross-framework definition of a blockchain node.
- Resiliency requirements—The same degree of security, decentralization, and replication must be required across multiple frameworks. This is particularly true for the execution of smart contracts and participation in the consensus protocol.
- The number of ledgers—It is necessary to fix the number of separate ledgers managed by each blockchain system and the workload to which each ledger is subject. Deploying multiple ledgers is an easy method to increase the throughput of a blockchain system.
- Standardized workloads—A suite of standardized tests must be designed. This allows for the assessment of an upper bound of the performance of a generic production system. Standardized workloads are also used by Hyperledger Caliper and the DLPS.

Concerning the functional requirements of nodes, Fabric differentiates among endorsing nodes, ordering nodes, and validating nodes. Similarly, Sawtooth separates the layers for ledger management, consensus protocol, and smart contract processing. However, a single Quorum node performs all three operations. Consequently, we provide an abstract definition of a blockchain node in terms of its functional requirements, which allows assigning framework modules to hardware resources consistently across the frameworks. An abstract blockchain node represents a non-trusting entity in a blockchain network and should perform all the relevant operations autonomously. We define an abstract blockchain node as a set of components performing all of the following tasks:

- peering and networking management (i.e., networking with other peers);
- consensus management (e.g., mining, fork resolution);
- transaction management and smart contract execution, which includes ordering, scheduling, and processing transactions;
- database management, which includes updating the ledger and the state database;
- security management, which includes cryptographic operations and privacy management.

We created networks of four nodes with each framework. We assigned four virtual machines to each network. Thus, we assigned a different virtual machine to each node. For each framework, we assigned modules to virtual machines to comply with the functional requirements of our definition of an abstract blockchain node, as shown in Fig. 3 and Table 2. Our tests do not involve private transactions. Thus, a single Besu or GoQuorum node satisfies the definition of an abstract blockchain node. Sawtooth offers many modules, as discussed in Section 2.7.2. In particular, a Sawtooth validator must always be connected to the transaction processor managing the on-chain settings. We also deployed the consensus engine and the transaction processor managing the transactions of our tests on the same virtual machine. We created four Fabric

Table 2
Test environment for each blockchain framework, highlighting the main differences among the various frameworks.

	Fabric	Sawtooth	Besu	GoQuorum
Version	2.2.2 (Jan 2021)	1.2.3 (Oct 2019)	21.1 (Feb 2021)	21.1 (Feb 2021)
Components per instance	1 peer, 1 orderer, 1 Fabric Certificate Authority	1 validator, 1 consensus engine, 1 REST API, 1 settings transaction processor, 1 test contract transaction processor	1 Besu node	1 GoQuorum node
Consensus	Raft	Raft, PBFT	IBFT 2.0	Raft, IBFT 1.0
Smart Contract	Go, Java	Go	Solidity	Solidity
State Database	LevelDB	LMDB	RocksDB	LevelDB
Batch Size	–	1 transaction	–	–
Endorsement Policy	All peers must endorse each transaction	–	–	–
Number of channels	1	–	–	–

organizations, as organizations in Fabric represent non-trusting parties. We deployed one peer and one orderer on a single virtual machine for each organization, as an abstract node must handle both transaction processing and consensus management.

Concerning the geographic distribution, for each framework, all the nodes were fully connected and deployed in the same VPC.

Concerning the resiliency requirements, all the nodes must participate in the consensus and execute each transaction. Consequently, in Fabric, transactions must be endorsed by all four nodes, as we want to execute each transaction exactly four times, once per node. When possible, we used equivalent consensus algorithms across the frameworks: Raft is implemented in all of the frameworks, whereas PBFT and IBFT behave similarly when consensus nodes are not dynamically replaced.

Concerning the number of ledgers, a single ledger was assigned to each blockchain system. Consequently, a single channel was used for Fabric.

For the workload simulation, the following scenarios were considered:

- the presence of parallelizable and sequential transactions—Sequential transactions are common when a process must be executed in steps, whereas parallel transactions are common when multiple independent processes occur at the same time, as in the case of sensors monitoring multiple assets;
- the presence of transactions writing a varying amount of data to the ledger—For example, this is common when using different IoT devices;
- the presence of transactions updating a varying number of objects—For example, a single sensor monitoring a cargo may need to update the data related to a single good or all the shipped goods simultaneously.

A single transaction type was defined for the performance evaluation. The transaction performed the following operations:

- loading a data structure from the ledger. The data structure contains a counter and a string;
- increasing the counter and replacing the string with its own payload;
- storing the data structure back to the ledger at its original address;
- repeating all previous steps for a certain number of iterations.

Consequently, each transaction was characterized by the following parameters:

- blockchain address, which is the location where the data structure is stored. When transactions target the same address, a sequential workload is generated. When transactions target different addresses, a parallelizable workload is generated;
- payload size, which specifies how much data are to be copied in the data structure modified by the transaction;
- number of iterations, which specifies how many times the transaction continues loading and storing data.

Three types of tests were performed on the frameworks, each of which focused on one of the aforementioned parameters:

- In the concurrency test, transactions read from and wrote to a varying number of different addresses. As such, it was possible to observe the behavior of the frameworks when transactions were sequential (i.e., they read from and wrote to the same address) or were parallelizable (i.e., they read from and wrote to completely different addresses).
- In the size test, transactions read from and wrote to the ledger a varying amount of data. As a single hash is usually no shorter than 0.1 kB, this value was used as the minimum payload size during the tests.
- In the iteration test, each transaction performed a varying number of load and store operations. This was used to simulate transactions by updating the state of one or more assets.

Table 3 summarizes the configuration used in each test. Each test was repeated ten times with each set of parameters. Transactions were submitted to one of the four nodes at a rate of 500 TPS (transactions per second). We chose such an input rate because it is higher than the maximum throughput reached by the frameworks, thus allowing us to highlight the different behaviors of the frameworks under the same workload. The performance was measured by a client external to the blockchain system under test. Consequently, time was measured by the client from transaction submission to transaction consolidation. Transactions were consolidated after a single block confirmation, as we used deterministic consensus algorithms. We underline that our objective is to measure the performance of the various frameworks under similar conditions. The results we obtained do not represent the maximum throughput of the frameworks. Measuring the maximum throughput would require solving a multi-dimensional maximization problem. Such problems are often non-polynomial and rarely solved exactly [83–85].

5.3. Environmental similarities and limitations

The frameworks were tested on the same hardware. Moreover, the configurations of the frameworks were not tuned. Depending on the programming languages supported by each framework, similar smart contracts were written in Go, Java, and Solidity. However, some differences existed due to the unique APIs offered by each framework. The main differences between the frameworks are reported in Table 2. For each framework, the table describes the version, the components instantiated on each virtual machine, the consensus protocol, the programming language used to implement the smart contracts, the default state database, the batch size (for Sawtooth), and the endorsement policy and the number of channels (for Fabric).

Table 3
Configuration of the parameters of the transactions for each type of test.

Test	No. addresses	Payload size (kB)	No. iterations
Concurrency	1, 100, max	0.1	1
Size	max	0.1, 1, 10, 20, 50	1
Iteration	max	0.1	1, 10, 100, 1000

In the concurrency test, the number of different addresses accessed ranged from one to the number of transactions submitted. In the size test, the payload size of the transactions ranged from 0.1 kB to 50 kB. In the iteration test, the number of read and write operations ranged from 1 to 1000.

5.4. Results

This section presents the results obtained from the performance evaluation.

The results of the concurrency test are shown in Fig. 4. Fabric did not perform well for sequential transactions: many of the transactions failed the validation step, as explained in Section 2. However, in the vast majority of use cases, transactions are parallelizable, and both Fabric and Quorum performed well. Sawtooth's performance was affected by the choice of small batches. In a similar test with larger batches [59], Sawtooth attained a TPS value half that achieved by Fabric. Moreover, in contrast to a previous study [59], Sawtooth's parallel scheduler did not provide any benefit. This was likely due to the choice of AWS instances with only two vCPUs. For Fabric, the choice of smart contract programming language was important, as those written in Java did not perform the same as those written in Go. CFT consensus algorithms boosted performance in all the frameworks, but on small networks, such as the one used for the tests, the performance gain did not justify the sacrifice of decentralization. However, by increasing the number of nodes, the performance advantages of using CFT algorithms on fully connected networks should become considerable, as they have lower message complexity. As the number of exchanged messages is relevant and not the total number of nodes, performances are unlikely to decay on large networks if each node is connected to a limited number of peers. This strategy is adopted by probabilistic consensus algorithms and impacts latency and finality instead of efficiency.

Fig. 5 presents the results of the size test, which confirmed the behaviors observed during the concurrency test. In addition, the performance of Besu rapidly decayed for heavier transactions. Overall, when increasing the size of the payload of the transactions, the TPS value decreased as the quantity of data stored per second increased.

Fig. 6 presents the results of the iteration test, which confirmed the performance decay of Besu under longer-lasting transactions. Overall, by increasing the number of load and store operations per transaction, the quantity of read and write operations per second increased, even if the TPS value decreased. Moreover, none of the frameworks seemed to be optimized for multiple read and write operations on the same address within the same transaction. In such cases, only the first read and the last write operations should be performed. This should be considered when writing smart contracts.

The results of our performance evaluation differ from those obtained by other studies. Such a condition is common to almost all the studies that use different tools, configurations, and testing methodologies. Thus, comparing our results to those in the literature is challenging. To limit such variability, we compare our results to those that use official versions of the frameworks. Moreover, we discard the studies that used outdated

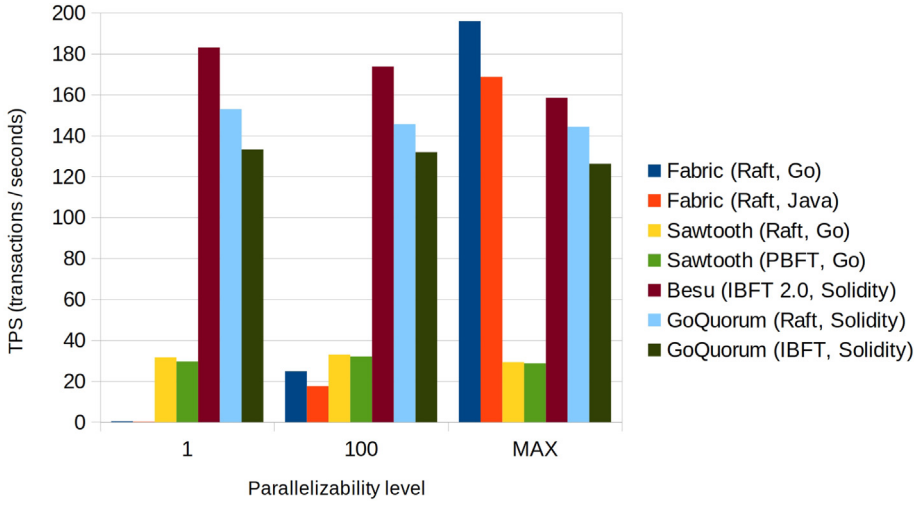

Fig. 4. Concurrency test: TPS for different levels of transaction parallelizability. Three scenarios were tested: sequential transactions (parallelizability = 1), partially parallelizable transactions (up to 100 parallel transactions), and independent transactions (max parallelizability).

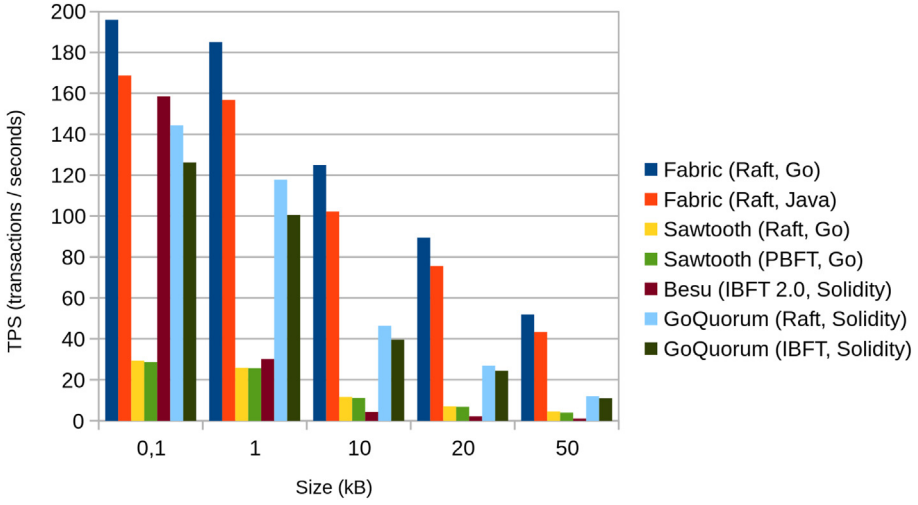

Fig. 5. Size test: TPS for different payload sizes. Various sizes were tested, ranging from 0.1 kB (approximately a 128-bit hash) to 50 kB.

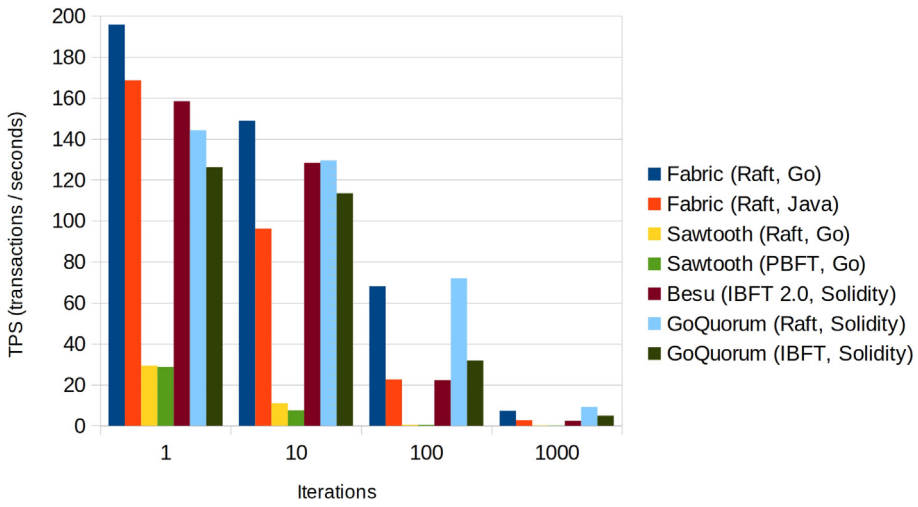

Fig. 6. Iteration test: TPS for different read/write amounts. A pair of read/write operations (iteration = 1) represents an update on a single asset. A set of multiple read/write operations represents a transaction updating multiple assets.

Table 4
Results of the performance evaluation.

No. addresses	Payload size (kB)	No. iterations	Fabric (Raft, Go)	Fabric (Raft, Java)	Sawtooth (Raft, Go)	Sawtooth (PBFT, Go)	Besu (IBFT 2.0, Solidity)	GoQuorum (Raft, Solidity)	GoQuorum (IBFT, Solidity)
1	0.1	1	$(3.0 \pm 0.2) \times 10^{-1}$	$(1.7 \pm 0.3) \times 10^{-1}$	$(3.1 \pm 0.1) \times 10$	$(2.9 \pm 0.1) \times 10$	$(1.8 \pm 0.3) \times 10^{2}$	$(1.2 \pm 0.03) \times 10^{2}$	$(1.33 \pm 0.07) \times 10^{2}$
100	0.1	1	$(2.4 \pm 0.2) \times 10$	$(1.7 \pm 0.3) \times 10$	$(3.2 \pm 0.1) \times 10$	$(3.2 \pm 0.5) \times 10$	$(1.7 \pm 0.4) \times 10^{2}$	$(1.46 \pm 0.05) \times 10^{2}$	$(1.32 \pm 0.05) \times 10^{2}$
max	0.1	1	$(1.95 \pm 0.02) \times 10^{2}$	$(1.68 \pm 0.02) \times 10^{2}$	$(2.9 \pm 0.2) \times 10$	$(2.8 \pm 0.4) \times 10$	$(1.6 \pm 0.5) \times 10^{2}$	$(1.44 \pm 0.04) \times 10^{2}$	$(1.26 \pm 0.08) \times 10^{2}$
max	0.1	10	$(1.48 \pm 0.05) \times 10^{2}$	$(9.6 \pm 0.2) \times 10$	$(1.0 \pm 0.1) \times 10$	(7 ± 1)	$(1.3 \pm 0.2) \times 10^{2}$	$(1.30 \pm 0.03) \times 10^{2}$	$(1.13 \pm 0.05) \times 10^{2}$
max	0.1	100	$(6.8 \pm 0.6) \times 10$	$(2.25 \pm 0.93) \times 10$	$(4.07 \pm 0.02) \times 10^{-1}$	$(3.7 \pm 0.1) \times 10^{-1}$	$(2.2 \pm 0.3) \times 10$	$(7.2 \pm 0.3) \times 10$	$(3.18 \pm 0.02) \times 10$
max	0.1	1000	(7.2 ± 0.1)	(2.63 ± 0.05)	$(1.26 \pm 0.02) \times 10^{-1}$	$(1.12 \pm 0.03) \times 10^{-1}$	(2.3 ± 0.1)	(9 ± 1)	(4.9 ± 0.5)
max	1	1	$(1.85 \pm 0.02) \times 10^{2}$	$(1.56 \pm 0.02) \times 10^{2}$	$(2.5 \pm 0.1) \times 10$	$(2.6 \pm 0.2) \times 10$	$(3.0 \pm 0.6) \times 10$	$(1.18 \pm 0.06) \times 10^{2}$	$(1.01 \pm 0.07) \times 10^{2}$
max	10	1	$(1.24 \pm 0.03) \times 10^{2}$	$(1.02 \pm 0.03) \times 10^{2}$	$(1.1 \pm 0.8) \times 10$	$(1.1 \pm 0.4) \times 10$	(4 ± 1)	$(5 \pm 2) \times 10$	$(4 \pm 1) \times 10$
max	20	1	$(8.9 \pm 0.3) \times 10$	$(7.5 \pm 0.2) \times 10$	(7 ± 5)	(7 ± 2)	(2.2 ± 0.5)	$(2.6 \pm 0.8) \times 10$	$(2 \pm 1) \times 10$
max	50	1	$(5.1 \pm 0.5) \times 10$	$(4.3 \pm 0.1) \times 10$	(4.6 ± 0.5)	(4 ± 1)	(1.1 ± 0.5)	$(1.2 \pm 0.4) \times 10$	$(1.1 \pm 0.5) \times 10$

Each row represents one of the tests performed. For each test, the configuration used and the results obtained are reported.

versions, as technological evolution may cause important differences in the measures.

Sedlmeir et al. [58] obtained much better results in terms of transaction throughput across all the frameworks. However, the author used more performing hardware. Moreover, even slight differences in the configuration of the frameworks may have a huge impact on the performance of the system. For example, we noticed that the performance of GoQuorum doubles when logging is disabled.

Guggenberger et al. [56] focused on Fabric only and used a different testing methodology based on an adaptive strategy that tries to match the input transaction rate to the output transaction rate. We believe such a strategy cannot be employed in a cross-chain comparison, as different frameworks would be subject to different input workloads. We prefer to use the same workload for different frameworks. Moreover, the authors used eight peers instead of the four we used. Thus, when four endorsers are busy validating a transaction, the other four can execute a different one, which doubles the overall throughput even when both studies use the same endorsement policy. Thus, the numerical values of the two studies are different. Nonetheless, there are some similarities in the overall behavior of the frameworks. In particular, the performance decay follows a similar pattern when the payload size increases.

Mazzoni et al. [6] did not provide information on the version of Quorum used in their experiments. Nonetheless, their paper was published recently. As the authors used Caliper to conduct their experiments on Quorum, transactions have a different complexity compared to ours. Moreover, the authors used a single virtual machine. Nonetheless, such a machine is more performing than the combined four used by us. Thus, even if some results may seem consistent between the two papers (e.g., 4-node Raft and 4-node IBFT), there are profound differences in the testing methodologies that prevent generalizations.

The obtained results are also provided in Table 4. Each row of the table represents one of the tests performed. For each test, the table reports the configuration used and the results obtained.

6. Conclusions and future developments

Blockchain is a rapidly evolving technology that has attracted the interest of many companies. However, many blockchain frameworks have emerged in the last few years. As such, choosing the most suitable framework is often a challenging task due to the general lack of updated

comparative analyses. In this study, after explaining why blockchain is important to the industry and why not all blockchains are equal, we focused on the following blockchain frameworks: Hyperledger Fabric v2.2.2, Hyperledger Sawtooth v1.2.3, and ConsenSys Quorum (with both the GoQuorum v21.1 client and the Hyperledger Besu v21.1 client). In particular, we performed a comparative analysis and evaluated the performance of the frameworks. Our findings can be used as a general reference for the industry. Overall, Fabric is efficient but lacks a BFT consensus algorithm; Sawtooth is flexible but not as efficient; and finally, Quorum performs well, offers a BFT consensus algorithm, and supports private transactions.

Future work will be aimed at improving the test methodology to overcome some of the limitations of the one proposed in this paper. For example, latency and read throughput could be included among the metrics to monitor. Moreover, as new frameworks emerge, similar analyses will need to be performed to provide a clear view of the blockchain landscape for both the industrial and academic worlds.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This study was partially supported by TIM in its Research agreement 2019–2021 with Politecnico di Torino.

While working on this paper, Prof. Guido Perboli was the head of the Urban Mobility and Logistics Systems initiative of the CARS@POLITO Interdepartmental Center.

References

[1] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008. (Accessed 20 April 2022).
[2] G. Perboli, S. Musso, M. Rosano, Blockchain in logistics and supply chain: a lean approach for designing real-world use cases, IEEE Access 6 (2018) 62018–62028, https://doi.org/10.1109/ACCESS.2018.2875782.
[3] K. Wust, A. Gervais, Do you need a blockchain?, in: Proceedings - 2018 Crypto Valley Conference on Blockchain Technology, CVCBT 2018, IEEE, 2018, pp. 45–54, https://doi.org/10.1109/CVCBT.2018.00011.
[4] T.G. Crainic, G. Perboli, M. Rosano, Simulation of intermodal freight transportation systems: a taxonomy, Eur. J. Oper. Res. 270 (2) (2018) 401–418, https://doi.org/10.1016/j.ejor.2017.11.061.
[5] G. Perboli, M. Rosano, A taxonomic analysis of smart city projects in north America and Europe, Sustainability 12 (18) (2020) 7813, https://doi.org/10.3390/su12187813.
[6] M. Mazzoni, A. Corradi, V. Di Nicola, Performance evaluation of permissioned blockchains for financial applications: the consensys quorum case study, Blockchain: Res. Appl. 3 (1) (2022) 100026, https://doi.org/10.1016/j.bcra.2021.100026.
[7] E. Fadda, G. Perboli, R. Tadei, Customized multi-period stochastic assignment problem for social engagement and opportunistic IoT, Comput. Oper. Res. 93 (2018) 41–50, https://doi.org/10.1016/j.cor.2018.01.010.
[8] S. Musso, G. Perboli, M. Rosano, A. Manfredi, A decentralized marketplace for M2M economy for smart cities, in: Proceedings of the 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), IEEE, 2019, pp. 27–30, https://doi.org/10.1109/WETICE.2019.00014.
[9] S. Pan, W. Zhou, S. Piramuthu, V. Giannikas, C. Chen, Smart city for sustainable urban freight logistics, Int. J. Prod. Res. 59 (7) (2021) 2079–2089, https://doi.org/10.1080/00207543.2021.1893970.
[10] G. Chiaselotti, T. Gentile, F. Infusino, Lattice representation with algebraic granular computing methods, Electron. J. Combinator 27 (1) (2020) 1–34, https://doi.org/10.37236/8786.
[11] E. Androulaki, A. Barger, V. Bortnikov, et al., Hyperledger fabric: a distributed operating system for permissioned blockchains, in: Proceedings of the Thirteenth EuroSys Conference, ACM, 2018, pp. 1–15. https://doi.org/10.1145/3190508.3190538.
[12] K. Olson, M. Bowman, J. Mitchell, et al., Sawtooth: an introduction. https://www.hyperledger.org/wp-content/uploads/2018/01/Hyperledger_Sawtooth_WhitePaper.pdf, 2018. (Accessed 20 April 2022).
[13] ConsenSys, Build on quorum, the complete open source blockchain platform for business. https://consensys.net/quorum/, 2021. (Accessed 20 April 2022).
[14] G. Perboli, V. Capocasale, D. Gotta, Blockchain-based transaction management in smart logistics: a sawtooth framework, in: Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference, COMPSAC, IEEE, 2020, pp. 1713–1718, https://doi.org/10.1109/COMPSAC48688.2020.000-8.
[15] Come-from-Beyond, Decentralized vs distributed, or why DLT is (probably) an incorrect term. https://medium.com/@comefrombeyond/decentralized-vs-distributed-or-why-dlt-is-probably-an-incorrect-term-fccbf62bdfe7, 2020. (Accessed 20 April 2022).
[16] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, H. Wang, Blockchain challenges and opportunities: a survey, Int. J. Web Grid Serv. 14 (2018) 352–375, https://doi.org/10.1504/IJWGS.2018.095647.
[17] R.v. Pelt, S. Jansen, D. Baars, S. Overbeek, Defining blockchain governance: a framework for analysis and comparison, Inf. Syst. Manag. 38 (1) (2021) 21–41, https://doi.org/10.1080/10580530.2020.1720046.
[18] V. Buterin, On public and private blockchains. https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains, 2015. (Accessed 20 April 2022).
[19] I.-C. Lin, T.-C. Liao, A survey of blockchain security issues and challenges, Int. J. Netw. Secur. 19 (5) (2017) 653–659, https://doi.org/10.6633/IJNS.201709.19(5).01.
[20] E. Olszewski, Why blockchain matters to enterprise (hint: it's not because of decentralization). https://medium.com/@eolszewski/why-blockchain-matters-to-enterprise-hint-its-not-because-of-decentralization-8c38674f43c6, 2019. (Accessed 20 April 2022).
[21] A. Baliga, Understanding blockchain consensus models. https://www.persistent.com/wp-content/uploads/2017/04/WP-Understanding-Blockchain-Consensus-Models.pdf, 2017. (Accessed 20 April 2022).
[22] Y. Xiao, N. Zhang, W. Lou, et al., A survey of distributed consensus protocols for blockchain networks, IEEE Commun. Surv. Tutorial. 22 (2) (2020) 1432–1465, https://doi.org/10.1109/COMST.2020.2969706.
[23] D. Ongaro, J. Ousterhout, In search of an understandable consensus algorithm, in: Proceedings of the 2014 USENIX Annual Technical Conference, USENIX, 2014, pp. 305–319.
[24] Hyperledger, Hyperledger sawtooth blockchain security (part one). https://www.hyperledger.org/blog/2018/11/09/hyperledger-sawtooth-blockchain-security-part-one, 2018. (Accessed 20 April 2022).
[25] G. Wood, Ethereum: a secure decentralised generalised transaction ledger. https://ethereum.github.io/yellowpaper/paper.pdf, 2021. (Accessed 20 April 2022).
[26] P. Szilágyi, Eip-225: Clique proof-of-authority consensus protocol. https://eips.ethereum.org/EIPS/eip-225, 2017. (Accessed 20 April 2022).
[27] M. Castro, B. Liskov, Practical Byzantine fault tolerance. Proceedings of the Third Symposium on Operating Systems Design and Implementation vol. 99, USENIX, 1999, pp. 173–186.
[28] ConsenSys, Scaling consensus for enterprise: explaining the IBFT algorithm. https://consensys.net/blog/enterprise-blockchain/scaling-consensus-for-enterprise-explaining-the-ibft-algorithm/, 2018. (Accessed 20 April 2022).
[29] V. Capocasale, G. Perboli, Standardizing smart contracts, IEEE Access 10 (2022) 91203–91212, https://doi.org/10.1109/ACCESS.2022.3202550.
[30] N. Szabo, Formalizing and securing relationships on public networks, First Monday. 2 (9) (1997), https://doi.org/10.5210/fm.v2i9.548.
[31] P. Sanz Bayón, Key legal issues surrounding smart contract applications, KLRI J. Law Legislat. 9 (1) (2019) 63–91, https://doi.org/10.2139/ssrn.3525778.
[32] Hyperledger blockchain performance metrics. https://www.hyperledger.org/resources/publications/blockchain-performance-metrics, 2019. (Accessed 20 April 2022).
[33] About hyperledger Foundation. https://www.hyperledger.org/about, 2020. (Accessed 20 April 2022).
[34] Hyperledger, A blockchain platform for the enterprise. https://hyperledger-fabric.readthedocs.io/en/release-2.2/, 2020. (Accessed 20 April 2022).
[35] ConsenSys, GoQuorum enterprise Ethereum client. https://docs.goquorum.consensys.net/en/stable/, 2020. (Accessed 20 April 2022).
[36] Hyperledger Besu community, Besu enterprise Ethereum client. https://besu.hyperledger.org/en/stable/, 2021. (Accessed 20 April 2022).
[37] D.P. Mera Quorum blockchain stress evaluation in different environments, Student Thesis, City University of New York, New York, NY, 2019.
[38] P. Thakkar, S. Nathan, B. Viswanathan, Performance benchmarking and optimizing hyperledger fabric blockchain platform, in: Proceedings of the 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE, 2018, pp. 264–276. https://doi.org/10.1109/MASCOTS.2018.00034.
[39] C. Gorenflo, S. Lee, L. Golab, S. Keshav, Fastfabric: scaling hyperledger fabric to 20 000 transactions per second, Int. J. Netw. Manag. 30 (5) (2020) e2099, https://doi.org/10.1002/nem.2099.
[40] J. Sousa, A. Bessani, M. Vukolic, A Byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform, in: Proceedings of the 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2018, pp. 51–58. https://doi.org/10.1109/DSN.2018.00018.
[41] A. Baliga, I. Subhod, P. Kamat, S. Chatterjee, Performance Evaluation of the Quorum Blockchain Platform, 2018 arXiv preprint arXiv:1809.03421.
[42] T. Dinh, J. Wang, G. Chen, R. Liu, B. Ooi, K.-L. Tan, Blockbench: a framework for analyzing private blockchains, in: Proceedings of the ACM SIGMOD International Conference on Management of Data F127746, ACM, 2017, pp. 1085–1100, https://doi.org/10.1145/3035918.3064033.
[43] S. Pongnumkul, C. Siripanpornchana, S. Thajchayapong, Performance analysis of private blockchain platforms in varying workloads, in: 2017 26th International Conference on Computer Communications and Networks, ICCCN 2017, IEEE, 2017, pp. 1–6, https://doi.org/10.1109/ICCCN.2017.8038517.

[44] Q. Nasir, I. Qasse, M. Abu Talib, A. Nassif, Performance Analysis of Hyperledger Fabric Platforms, Secur. Comm. Netw. 2018 (2018) 3976093, https://doi.org/10.1155/2018/3976093.
[45] A. Sharma, D. Agrawal, F. Schuhknecht, J. Dittrich, Blurring the lines between blockchains and database systems: the case of hyperledger fabric, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM, 2019, pp. 105–122, https://doi.org/10.1145/3299869.3319883.
[46] S. Shalaby, A. Abdellatif, A. Al-Ali, A. Mohamed, A. Erbad, M. Guizani, Performance evaluation of hyperledger fabric, in: 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies, ICIoT 2020, IEEE, 2020, pp. 608–613, https://doi.org/10.1109/ICIoT48696.2020.9089614.
[47] H. Sukhwani, N. Wang, K. Trivedi, A. Rindos, Performance modeling of hyperledger fabric (permissioned blockchain network), in: NCA 2018 - Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications, IEEE, 2018, pp. 1–8, https://doi.org/10.1109/NCA.2018.8548070.
[48] B. Ampel, M. Patton, H. Chen, Performance modeling of hyperledger sawtooth blockchain, in: Proceedings of the 2019 IEEE International Conference on Intelligence and Security Informatics, ISI 2019, IEEE, 2019, pp. 59–61, https://doi.org/10.1109/ISI.2019.8823238.
[49] T. Sund, C. Lööf, S. Nadjm-Tehrani, M. Asplund, Blockchain-based event processing in supply chains—a case study at ikea, Robot. Comput. Integrated Manuf. 65 (2020) 101971, https://doi.org/10.1016/j.rcim.2020.101971.
[50] A. Corso, Performance Analysis of Proof-Of-Elapsed-Time (Poet) Consensus in the Sawtooth Blockchain Framework, Ph.D. thesis, University of Oregon, Eugene, OR, 2019.
[51] M. Rasolroveicy, M. Fokaefs, Performance evaluation of distributed ledger technologies for iot data registry: a comparative study, in: Proceedings of the World Conference on Smart Trends in Systems, Security and Sustainability, WS4 2020, IEEE, 2020, pp. 137–144, https://doi.org/10.1109/WorldS450073.2020.9210358.
[52] S. Benahmed, I. Pidikseev, R. Hussain, J. Lee, S. Kazmi, A. Oracevic, F. Hussain, A comparative analysis of distributed ledger technologies for smart contract development, in: Proceedings of the 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, IEEE, 2019, pp. 1–6. http://doi.org/10.1109/PIMRC.2019.8904256.
[53] T. Nakaike, Q. Zhang, Y. Ueda, T. Inagaki, M. Ohara, Hyperledger fabric performance characterization and optimization using goleveldb benchmark, in: Proceedings of the 2020 IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2020, vol. IEEE, 2020, pp. 1–9. http://doi.org/10.1109/ICBC48266.2020.9169454.
[54] C. Wang, X. Chu, Performance characterization and bottleneck analysis of hyperledger fabric, in: Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems, IEEE, 2020, pp. 1281–1286, https://doi.org/10.1109/ICDCS47774.2020.00165.
[55] Z. Shi, H. Zhou, Y. Hu, S. Jayachander, C. De Laat, Z. Zhao, Operating permissioned blockchain in clouds: a performance study of hyperledger sawtooth, in: Proceedings - 2019 18th International Symposium on Parallel and Distributed Computing, ISPDC 2019, IEEE, 2019, pp. 50–57, https://doi.org/10.1109/ISPDC.2019.00010.
[56] T. Guggenberger, J. Sedlmeir, G. Fridgen, A. Luckow, An in-depth investigation of the performance characteristics of hyperledger fabric, Comput. Ind. Eng. 173 (2022), 108716, https://doi.org/10.1016/j.cie.2022.108716.
[57] M. Kwon, H. Yu, Performance improvement of ordering and endorsement phase in hyperledger fabric, in: 2019 6th International Conference on Internet of Things: Systems, Management and Security, IOTSMS 2019, IEEE, 2019, pp. 428–432, https://doi.org/10.1109/IOTSMS48152.2019.8939202.
[58] J. Sedlmeir, P. Ross, A. Luckow, J. Lockl, D. Miehle, G. Fridgen, The Dlps: A New Framework for Benchmarking Blockchains, Proceedings of the 54th Hawaii International Conference on System Sciences, HICSS, 2021, pp. 1–10.
[59] V. Capocasale, D. Gotta, S. Musso, G. Perboli, A blockchain, 5g and iot-based transaction management system for smart logistics: an hyperledger framework, in: 2021 IEEE 45th Annual Computers, Software, and Applications Conference, COMPSAC, IEEE, 2021, pp. 1285–1290, https://doi.org/10.1109/COMPSAC51774.2021.00179.
[60] J. Polge, J. Robert, Y. Le Traon, Permissioned blockchain frameworks in the industry: a comparison, ICT Express 7 (2021) 229–233, https://doi.org/10.1016/j.icte.2020.09.002.
[61] A. Monrat, O. Schelen, K. Andersson, Performance evaluation of permissioned blockchain platforms, in: Proceedings of the 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering, CSDE, IEEE, 2020, pp. 1–8, https://doi.org/10.1109/CSDE50874.2020.9411380.
[62] Hyperledger Caliper, Getting started. https://hyperledger.github.io/caliper/v0.5.0/getting-started/, 2022. (Accessed 20 April 2022).
[63] D. Saingre, T. Ledoux, J.-M. Menaud, Bctmark, A framework for benchmarking blockchain technologies, in: Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA, IEEE, 2020, pp. 1–8, https://doi.org/10.1109/AICCSA50499.2020.9316536.
[64] J. Shah, D. Sharma, Performance benchmarking frameworks for distributed ledger technologies, in: Proceedings of CONECCT 2021: 7th IEEE International Conference on Electronics, Computing and Communication Technologies, IEEE, 2021, pp. 1–5, https://doi.org/10.1109/CONECCT52877.2021.9622659.
[65] G. Shapiro, C. Natoli, V. Gramoli, The performance of byzantine fault tolerant blockchains, in: 2020 IEEE 19th International Symposium on Network Computing and Applications, NCA, IEEE, 2020, pp. 1–8, https://doi.org/10.1109/NCA51143.2020.9306742.
[66] H. Moog, A new "consensus": the tangle multiverse [part 1]. https://husqy.medium.com/a-new-consensus-the-tangle-multiverse-part-1-da4cb2a69772, 2019. (Accessed 20 April 2022).
[67] J. Ma, Y. Jo, C. Park, Peerbft: making hyperledger fabric's ordering service withstand byzantine faults, IEEE Access 8 (2020) 217255–217267, https://doi.org/10.1109/ACCESS.2020.3040443.
[68] Intel Corporation, Introduction. https://sawtooth.hyperledger.org/docs/core/releases/1.2.6/introduction.html, 2017. (Accessed 20 April 2022).
[69] M. del Castillo, Blockchain 50 2021. https://www.forbes.com/sites/michaeldelcastillo/2021/02/02/blockchain-50/, 2021. (Accessed 20 April 2022).
[70] Linux Foundation, Case studies. https://www.hyperledger.org/learn/case-studies, 2020. (Accessed 20 April 2022).
[71] M. Caro, M. Ali, M. Vecchio, R. Giaffreda, Blockchain-based traceability in agri-food supply chain management: a practical implementation, in: Proceedings of the 2018 IoT Vertical and Topical Summit on Agriculture - Tuscany, IOT Tuscany 2018, IEEE, 2018, pp. 1–4, https://doi.org/10.1109/IOT-TUSCANY.2018.8373021.
[72] D. Bumblauskas, A. Mann, B. Dugan, J. Rittmer, A blockchain use case in food distribution: do you know where your food has been? Int. J. Inf. Manag. 52 (2020) 102008, https://doi.org/10.1016/j.ijinfomgt.2019.09.004.
[73] A.A. Khan, A.A. Laghari, D.-S. Liu, A.A. Shaikh, D.-A. Ma, C.-Y. Wang, A.A. Wagan, Eps-ledger: blockchain hyperledger sawtooth-enabled distributed power systems chain of operation and control node privacy and security, Electronics 10 (19) (2021) 2395, https://doi.org/10.3390/electronics10192395.
[74] Chainstack, Enterprise blockchain protocols evolution index 2020. https://chainstack.com/resources/#enterprise-blockchain-protocols-evolution-index-2020, 2020. (Accessed 20 April 2022).
[75] Chainstack, Enterprise blockchain protocols evolution index. https://chainstack.com/download/enterprise-blockchain-protocols-evolution-index-2021/, 2021. (Accessed 20 April 2022).
[76] S. Motepalli, H.-A. Jacobsen, Decentralizing Permissioned Blockchain with Delay Towers, 2022 arXiv preprint arXiv:2203.09714.
[77] S. Wang, M. Yang, Y. Zhang, Y. Luo, T. Ge, X. Fu, W. Zhao, On private data collection of hyperledger fabric, in: Proceedings - International Conference on Distributed Computing Systems, IEEE, 2021, pp. 819–829, https://doi.org/10.1109/ICDCS51616.2021.00083.
[78] N. Adarme, Tessera: the privacy manager of choice for ConsenSys Quorum networks. https://consensys.net/blog/quorum/tessera-the-privacy-manager-of-choice-for-consensys-quorum-networks/, 2021. (Accessed 20 April 2022).
[79] P. Lafourcade, M. Lombard-Platet, About blockchain interoperability, Inf. Process. Lett. 161 (2020) 105976, https://doi.org/10.1016/j.ipl.2020.105976.
[80] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, W. Knottenbelt, in: N. Borisov, C. Diaz (Eds.), Financial Cryptography and Data Security, Springer, Berlin, Heidelberg, 2021, pp. 3–36, https://doi.org/10.1007/978-3-662-64331-0_1.
[81] R. Belchior, A. Vasconcelos, S. Guerreiro, M. Correia, A survey on blockchain interoperability: past, present, and future trends, ACM Comput. Surv. 54 (2022), https://doi.org/10.1145/3471140.
[82] M. Battagliola, A. Flamini, R. Longo, et al., Quadrans blockchain. https://quadrans.io/content/files/quadrans-yellow-paper-rev02.pdf, 2021. (Accessed 20 April 2022).
[83] T.G. Crainic, G. Perboli, R. Tadei, Recent advances in multidimensional packing problems, in: C. Volosencu (Eds.), New Technologies–Trends, Innovations and Research, IntechOpen Limited, London, 2012, pp. 91–110, https://doi.org/10.5772/33302.
[84] M. Boccia, A. Mancuso, A. Masone, C. Sterle, A feature based solution approach for the flying sidekick traveling salesman problem, in: A. Strekalovsky, Y. Kochetov, T. Gruzdeva, A. Orlov (Eds.), Mathematical Optimization Theory and Operations Research: Recent Trends, Springer International Publishing, Cham, 2021, pp. 131–146, https://doi.org/10.1007/978-3-030-86433-0_9.
[85] G. Caselli, D. De Santis, M. Delorme, M. Iori, A mathematical formulation for reducing overcrowding in hospitals' waiting rooms, in: 2021 IEEE International Conference on Industrial Engineering and Engineering Management, IEEM 2021, IEEE, 2021, pp. 297–301, https://doi.org/10.1109/IEEM50564.2021.9673050.