How To Benchmark Permissioned Blockchains

Jeeta Ann Chacko[1], Ruben Mayer[2], Alan Fekete[3], Vincent Gramoli[3], and Hans-Arno Jacobsen[4]

[1] Technical University of Munich
chacko@in.tum.de
[2] University of Bayreuth
ruben.mayer@uni-bayreuth.de
[3] University of Sydney
{alan.fekete,vincent.gramoli}@sydney.edu.au
[4] University of Toronto
jacobsen@eecg.toronto.edu

Abstract. Blockchain benchmarking systems are actively discussed in the literature, focusing on increasing the number of blockchains that can be supported. However, the constant inception of new blockchains into the market and their vast implementation differences make it a massive engineering challenge. We provide a general discussion on the main aspects of benchmarking blockchains, highlighting the necessary contributions from the developers and users of blockchains and benchmarking systems. We identify problem statements across four benchmarking factors by investigating five popular permissioned blockchains. Further, we define a broad methodology to tackle these problems. We conduct a case study of five existing blockchain benchmarking systems for our evaluation and identify their limitations, clarifying the need for our methodology.

Keywords: permissioned blockchains · benchmarking systems

1 Introduction

Though blockchains were initially considered digital currency exchange systems, introducing smart contracts led to the classification of blockchains as decentralized transactional management systems that could support more use cases [64]. Later, the conception of permissioned blockchains that restricted the network access to authorized users and improved the overall performance made blockchains attractive for enterprise use cases. Currently, the most popular permissioned blockchain platforms, such as Fabric, Corda, Multichain, and Quorum, have around 30 to 70 enterprise partners using their systems for various use cases, such as banking, supply chain transparency, and digital asset management [21, 43, 55, 60].

However, the plethora of blockchain systems currently available in the market creates uncertainty in the selection process. A recent survey shows that 26% of users switched from their initially chosen blockchain at a later stage of development and that performance is one of the top selection criteria for blockchains [63].

Though most blockchains report their individual performance data, the vast differences in implementation, system configuration, and workloads make a fair comparison challenging [34]. This highlights the demand for a comprehensive and impartial blockchain benchmarking approach. Currently, there are multiple benchmarking system implementations available for blockchains [12,29,34,41,58, 65]. Each of them targets one or a specific set of blockchains to benchmark. The current focus in this research space is on increasing the number of blockchains supported by a benchmarking system. For example, Blockbench [29], the first benchmarking system for permissioned blockchains, supports four blockchains, while Diablo [34] and Gromit [58], the latest benchmarking systems, support seven blockchains. However, the rapid inception of new blockchains into the market makes this a massive engineering challenge.

Additionally, as is the case with most transaction processing systems, initially, the lines between the design and implementation of benchmarking systems are often blurred [5]. A well-implemented benchmarking system may still fail to consider all crucial aspects of benchmarking due to poor design [35]. For example, many existing benchmarking systems only support simple asset transfer scenarios [58, 65], while in reality, blockchains are employed for numerous other use cases. Therefore, we identify the need for a thorough discussion on the different aspects of benchmarking blockchains, which will assist in implementing a comprehensive and extensible benchmarking system in the future.

One needs to understand the similarities and differences between the various blockchain platforms to identify the diverse factors of benchmarking accurately. A significant challenge in this direction is the insufficient scientific literature. Since many blockchains are commercialized, apart from research papers, we must also analyze technical documentation and blog posts from the respective blockchain developers to understand their systems thoroughly. Further, given the vast implementation distinctions among the different blockchain systems, discussions regarding blockchain benchmarking should not be limited to developers of benchmarking systems, but should also include developers of blockchain systems.

Our discussions address the problems regarding crucial benchmarking elements such as system configuration, parameter tuning, workloads, and metrics. We emphasize the importance of these issues by extensively analyzing five different permissioned blockchains. We then define the contributions required from the entire blockchain community to tackle them. We also conduct a case study of five existing benchmarking systems to identify their limitations and highlight the need for contributions. For example, we identify various system configuration settings that affect the performance of each of the multiple blockchains, while current benchmarking studies only employ the default value for these settings. In detail, we provide the following contributions:

1. We formulate problem statements across four aspects of benchmarking based on five different permissioned blockchains (Fabric, Corda, Multichain, Quorum and Diem). This highlights the importance of these problems across different blockchain platforms.

2. We define a general methodology to tackle the problems that spans across developers and users of blockchains as well as benchmarking systems. This highlights the contributions required from each of them to improve the domain of blockchain benchmarking.
3. We provide a case study of five different blockchain benchmarking systems and the corresponding benchmarking studies to highlight the current limitations. This can help benchmarking system developers to extend their implementations to adhere to our methodology.

2 Permissioned Blockchains

In permissioned blockchains, access is restricted to a set of authorized users, making them suitable for many enterprise use cases that cannot support anonymity. They are peer-to-peer networks with access controls operating on a distributed ledger. Despite being in the same classification, the multiple permissioned blockchain systems currently available have vast differences in their implementation. This section briefly overviews the basic concepts and transaction flow of five popular permissioned blockchains, accentuating their similarities and differences.

2.1 Hyperledger Fabric

Hyperledger Fabric (a.k.a Fabric) is an open-sourced, permissioned blockchain system under the Linux foundation [3]. Fabric follows an execute-order-validate (EOV) model, one of its unique features. The main components of a Fabric network are peers, endorsers, and the ordering service. Only the endorsers store the smart contracts, and transactions are sent to the endorsers for execution based on an endorsement policy. Speculative transaction execution results in a read-write set of all the keys in the transaction which is then forwarded to the ordering service. The ordering service is a cluster of nodes that employs the Raft consensus protocol to decide on the order of the transactions. Upon consensus, a block of ordered transactions is broadcasted to all peers. Every peer validates the speculative results of every transaction in the block with the current world state. After successful validation, the world state is updated, and the block of transactions is committed to the ledger.

2.2 Corda

R3 Corda is an open-sourced permissioned blockchain mainly designed for financial use cases [8]. In Corda, data is only shared among the network participants on a need-to-know basis, one of its unique features. The nodes in a Corda network are authorized using an identity service. Further, a network map service is employed for node lookup, enabling point-to-point communication between nodes. An immutable object called a state describes any data known to the nodes at a specific point in time. Each node has a vault or database that stores all the state sequences it knows. Constraints to ensure that a state is valid are defined using smart contracts. A transaction defines the input and output for a state transformation. Further legal prose can be attached to a transaction to settle future disputes, which makes Corda appealing to financial use cases. Notaries are specific nodes assigned with the responsibility of ensuring that output states are unique successors of input states thereby preventing double spending. When a transaction proposal is created, only the entities related to it execute the smart contract to ensure its validity. Further, notaries check each input state object in a transaction to ensure that they have not been consumed earlier and prevent double-spending. Transactions are committed after the transaction-related entities and the notaries sign them.

2.3 Multichain

Multichain is a fork of Bitcoin and shares many of its features [36]. However, it is designed for a permissioned environment where nodes prove their identity using a handshaking protocol when connecting to other nodes. Each node defines the public address for which it has a private key, and other nodes can send challenge messages to be signed with this key. Unlike Bitcoin, only a few nodes are granted mining privileges, and there is a single validator per block. The validator is scheduled in a round-robin style with tunable parameters. Other participants then execute the individual transactions in a block in the defined global order.

2.4 Quorum

Quorum is a fork of the Go implementation of Ethereum, where the P2P layer was redesigned to allow only authorized nodes [51]. A privacy layer is implemented to support private and public transactions in a permissioned environment. Quorum uses transaction managers to handle encrypted data, including an enclave, which is a hardware security module, to hold private keys. Private transactions are sent to transaction managers for encryption after verifying the sender, and only the entities related to the transaction can receive the decrypted data. Different protocols, such as Raft, IBFT, and QBFT, are employed to attain consensus in the Quorum network. When consensus is reached, all the nodes in the network execute the public transactions in a block, while private transactions are only executed by the entities related to the transaction.

2.5 Diem

Diem (earlier known as Libra) is a permissioned blockchain introduced by Facebook (now Meta) [28]. The network consists of two types of nodes - full nodes and validators. For every incoming transaction, validators check the signature, balance, and whether the transaction has been replayed, before sharing them with other validators. A BFT protocol (DiemBFT) is used to reach a consensus on the order of transactions. When a validator is elected as the leader, it proposes a block which is forwarded to the other validators for approval. Meanwhile, the transactions in the block are speculatively executed and also shared. Upon consensus, all the transactions of the proposed block are committed. Full nodes are employed to re-execute and store all transactions to provide evidence in the event of a history rewrite attempt. It ensures that validators cannot collude on transaction executions.

3 Benchmarking Guidelines

In this section, we focus on four important aspects to address when benchmarking permissioned blockchains. We consider examples from Fabric [3], Corda [38], Multichain [56], Quorum [61] (four of the most commonly used permissioned blockchains [63]) and Diem [28] when defining each problem statement. We then propose a general methodology for tackling each problem. The methodology targets blockchain developers, benchmarking system developers, as well as those conducting benchmarking studies on blockchains. We aim to bring to light the contributions required from each of them to the blockchain benchmarking space.

3.1 System Configuration

Problem Statement There is a vast distinction in the system components that compose the different permissioned blockchain implementations. The choice, count, and distribution of the different components significantly affect the performance. For example, the Hyperledger Fabric network consists of validating peers, endorsers, orderers, and clients where the count and distribution of each of these components impact the performance [10,13,39]. Corda offers two configurations for its notary nodes: validating and non-validating. Deploying multiple validating notary clusters can aid load balancing, improving performance [20]. In the Quorum network, performance is influenced by the choice between full nodes with a privacy manager or light nodes [33] for process-intensive tasks as well as boot nodes [15] or static nodes [16] for different peer discovery strategies. Multichain has the concept of data streams, and nodes that subscribe to these streams ensure faster information retrieval [53]. Also, Diem has the concept of validator nodes and full nodes, the choice of which can introduce additional overhead depending on the use case [27]. Therefore, identifying the influential system components and designing the optimal setup is crucial to ensure the best performance for each blockchain implementation. Further, even though the system configuration of each blockchain needs to be individually optimized, the hardware requirements or the hardware cost must be uniform across all blockchains for a fair benchmarking approach [62].

Methodology

1. Blockchain system developers need to provide extensive documentation and experimental results to quantify the influence of system components on the performance for each blockchain. Identifying and documenting a priority-based list of the main components that significantly impact the performance

will be highly beneficial. Multichain published a list of tips for performance optimization on their website [54] which includes ideal server specifications, and though they do not provide concrete suggestions, this highlights the need for such documentation from the developers.

2. Benchmarking system users must design an optimal system setup specific to each blockchain based on their documentation. This is a challenging yet crucial task. Individually optimizing the system setup ensures benchmarking the best performing setup of each blockchain. Further, all the blockchains benchmarked together must employ uniform hardware or be limited to uniform hardware costs to ensure fairness [62].
3. Benchmarking system developers must support easy integration and reconfiguration of all system components. Due to the large number and type of components involved, system setup is often complex for blockchains [71]. Benchmarking systems need to provide automation scripts or at least detailed documentation that supports the integration of influential system components apart from the default to ease the system setup process. Further, the optimal system setup varies with use cases, so easy reconfiguration should also be supported.

3.2 Parameter Tuning

Problem Statement Parameter tuning is a significant factor to consider while benchmarking blockchains, and it is heavily discussed in the literature [10,48,70]. The literature mainly discusses generic parameters, such as block size, while system-specific parameters are largely ignored. However, both are equally important to ensure fair benchmarking. The number of transactions to include in a block is a well-known parameter that influences the performance of most blockchains [10, 50], but Corda is an exception since the concept of blocks does not exist [19]. Further, there are system-specific parameters such as the set of cache-related parameters for GoQuorum [31, 32] and Corda [18], the validator pool size, endorsement policy, and CouchDB parameters for Fabric [13], or the mining diversity and skip proof-of-work check [52] configurations in Multichain, all of which can be tuned for performance improvements. Also, Diem offers mempool [26] and consensus [25] configurations that are based on its unique implementation. Therefore, individually identifying and tuning the critical parameters for each blockchain is required to benchmark the ideal performance of every system. However, on the other hand, some parameters may impact the system's functionality and must be set equivalently to ensure a fair comparison. For example, Clique is byzantine fault tolerant with eventual finality, while Raft is only crash fault tolerant with immediate finality, and either can be chosen as the consensus protocol in Quorum [17]. If Fabric, which offers only the Raft consensus protocol, is benchmarked with Quorum, then to ensure fairness, Quorum's consensus protocol needs to be set to Raft. Parameter tuning is often discussed in the literature on benchmarking transaction processing systems [5, 35]. However, when considering blockchains, there is a more diversified set of parameters for tuning since the blockchain stack is comprised of numerous layers such as consensus models, access control protocols, database stores, smart contracts, and distributed ledgers.

Methodology

1. Blockchain developers should identify key parameters that influence the performance of their blockchain. They should ensure that all configuration parameters and quantitative evidence of their effect on performance are well documented. Workload-based analysis of these parameters should also be conducted and documented. Further, given the large number of parameters in blockchains, a prioritizing strategy would be beneficial. For example, Fabric has over 50 parameters, and a recent study quantitatively ranked the top parameters that significantly affect the performance [47].
2. Benchmarking system users must tune parameters based on the workload and system setup. Currently, benchmarking is often accomplished with the default parameter values or with a one-time tuning of limited parameters [29, 34, 58]. However, studies show that parameter tuning significantly depends on the workload and system setup [10, 48, 70]. Therefore, parameter tuning should be done dependent on the use case that is being benchmarked. Recently, auto-tuning of blockchains is also being discussed in the literature, which could ease this process [47].

3.3 Workloads and Use Cases

Problem Statement The third important aspect to consider is the workload employed for benchmarking. Using existing workloads such as YCSB and TPC is a popular choice since these are well established in the community [29, 45]. However, blockchains often target different use cases than traditional transaction processing systems. Therefore, reusing existing workloads is often unrealistic and leads to inaccurate assumptions about the performance of a system [34]. Further, blockchain implementations are varied, and each is designed with a specific use case in mind. For example, Fabric cannot handle highly skewed workloads due to its optimistic concurrency control model [10], and Corda supports only point-to-point requests between entities involved in a transaction [19]. Also, Quorum and Corda are mainly popular for financial use cases while Fabric applications range across multiple domains such as supply chain management and healthcare [73]. Further, the system setup, parameters, and transaction definition also vary with the use case. Multichain recommends different performance optimization strategies based on the expected type of workload [54], and Diem defines three different types of transactions based on the client account type [72].

Methodology

1. Benchmarking system developers should focus on traditional as well as blockchain-specific workloads. Porting traditional workloads such as TPC and YCSB to blockchain environments is a good practice as it corresponds to scenarios

where existing enterprise applications are migrated to blockchain platforms. However, the focus should also be given to blockchain-specific workloads, such as supply chain and digital asset management scenarios, to capture realistic performance capabilities better. Apart from workload generation, converting or porting the workloads to support multiple blockchain implementations is an important and challenging engineering task.

2. Benchmarking system developers must also generate system-specific workloads. Such workloads that stress test distinctive blockchains based on their specific design are essential to highlight accurate performance expectations. For example, private transactions in Fabric and point-to-point requests in Corda would need specific workloads different from other generic broadcast transactions. Also, the targeted use cases of each blockchain implementation should be supported.
3. Benchmarking system users and blockchain developers should provide use case-based discussion of benchmarking results. Benchmarking results will quantitatively indicate the most or least performant blockchain. However, a specific blockchain's intended use case must be considered before reaching a viable conclusion. For example, it has been quantitatively shown that Fabric is more performant than Diem [76]. However, Diem supports a byzantine fault-tolerant consensus protocol, while Fabric uses a crash fault-tolerant consensus protocol, both of which are suitable for entirely different use cases. Therefore, evaluation results need to be explored extensively in relation to the blockchain implementation and envisioned use.

3.4 Performance Metrics

Problem Statement The metrics used for benchmarking depend on the quality being benchmarked. Throughput and latency are the main client-visible metrics generally used in benchmarking blockchains when the focus is on performance; as well, some studies look at metrics reported from the blockchain platform, such as CPU usage or storage. However, there needs to be more clarity about how to define these metrics. Throughput is often defined as the number of transactions committed to a blockchain per second. However, for Fabric, failed transactions are also committed to the blockchain [3]. Latency is often described as the duration between transaction submission and final commit. However, submission time can be considered as the time the client submitted the transaction or the time the transaction entered the consensus protocol [29, 46]. Further, depending on whether the blockchain supports immediate or probabilistic finality, the definition of commit time changes [58]. Also, latency is a distribution, and single summary values such as mean or 95-percentile can be quoted, depending on what matters most for the specific use case. Therefore, a uniform definition for blockchain performance metrics is challenging. Further, system-specific metrics also need to be considered to provide a better understanding for the client. For example, apart from throughput and latency, Diem developers define a metric called *capacity* as "*the ability of the blockchain to store a large number of accounts*" [2].

Methodology

1. Blockchain developers must define generic as well as system-specific performance metrics. Generic metrics should either be uniformly defined for all blockchains along with the system-specific assumptions or be uniquely defined for each blockchain (or both). System-specific performance metrics must be clearly defined, and the necessity for these metrics must also be clarified.
2. Benchmarking system developers should support fine-grained result generation. Since the metric definition varies for each blockchain, publishing all variations of a metric in the results will be helpful for better understanding. In most cases, simple mathematical calculations can provide more fine-grained results. For example, the results from the Caliper benchmarking system display only the "*success throughput*" and not the "*commit throughput*" even though both can be derived from the available results [10].

4 Case Study

In this section, we analyze five different benchmarking systems that support permissioned blockchains [6,9,23,37,41] as well as the corresponding five benchmarking studies conducted using these systems [10,29,34,58,65]. Our discussion is mainly based on the benchmarking studies as this is representative of how the benchmarking system is used in practice. Table 1 summarizes the integrated blockchain systems, available workloads, and published performance metrics for each benchmarking system. We intend to identify the limitations of the current benchmarking systems through this case study which can help develop a more comprehensive system.

Scope. We observe that none of the benchmarking systems currently support all four of the most commonly used permissioned blockchains. One of the main reasons for benchmarking is for clients to choose the appropriate blockchains based on their requirements. Therefore, a benchmarking system must support at least the most popular blockchain choices. However, the engineering challenge behind implementing such a comprehensive benchmarking system is immense. Alternatively, providing documentation that accurately details the exact procedure to integrate any new blockchain into an existing benchmarking system would be beneficial. Diablo, Gromit, and BCTMark provide short documentation or discussions on integrating new blockchains into their benchmarking system [24, 58, 65]. Caliper provides extensive documentation that details the steps required to implement a connector to integrate a new blockchain [75]. This includes the requirements of the connector, implementation, binding, and integration, as well as instructions on how to document the newly developed connector for future users. Despite the well-defined documentation, there has been little effort from the community to integrate more blockchains into Caliper.

System Configuration. The existing benchmarking systems support the evaluation of the different blockchains on scaling hardware configurations. Diablo, Gromit, and HyperledgerLab emulate geo-distribution. However, system

Table 1. Blockchain Benchmarking Systems

Benchmarking Systems	**Supported blockchains (permissioned underlined)**	**Supported Workloads**	**Performance Metrics**
Blockbench [29]	Ethereum [74], Fabric [3], Parity [59], Quorum [61]	YCSB, smallbank, etherId, doubler, wavesPresale, doNothing, analytics, IOHeavy, CPUHeavy [6]	success throughput, average latency
HyperledgerLab [10], Caliper [41]	Fabric, Ethereum, Besu [40]	simple asset transfer, smallbank, fabcar, synthetic generator, electronic health records, digital music management, e-voting, supply chain management [42, 44]	commit throughput, success throughput, average latency
Diablo [34]	Algorand [30], Avalanche [1], Ethereum, Diem [4], Solana [68], Quorum, RedBelly [22, 69]	exchange DApp, gaming DApp, webservice DApp, mobility service DApp, video sharing DApp [23]	throughput, average latency, proportion of committed transactions, peak transaction throughput, latency distribution over time
Gromit [58]	Ethereum, Algorand, BitShares [66], Diem, Fabric, Stellar [49], Avalanche	simple asset transfer [37]	peak transaction throughput, average latency
BCTMark [65]	Ethereum, Clique [14], Fabric	synthetic generator, history-based , sorting algorithms [9]	CPU usage, HDD usage, memory consumption, gas cost

configuration is not extensively evaluated in the corresponding benchmarking studies. The number of hardware nodes and, correspondingly, the number of peers in a system are scaled and evaluated but the peer configurations are kept constant. Currently, system configuration is considered independent from the benchmarking systems and is left to the client's responsibility. Providing automated testbed setups for the supported blockchains can ease the benchmarking effort with varying system configurations. The HyperledgerLab benchmarking system includes such an automated testbed and therefore can evaluate the effect of endorser and database configurations, but it is limited to Fabric.

Parameter Tuning. In the studies we examined, system parameters are mostly kept with the default value used in whichever blockchain is being tested. Blockbench tunes the *difficulty* variable for Ethereum to limit miners from diverging [29]. HyperledgerLab evaluates the effect of system parameters such as *block size* and *endorsement policy* but is limited to Fabric [10]. Tuning the parameters of individual blockchains to ensure the fair comparison of the best performance of all the systems under test is a massive challenge due to the large number of parameters involved. Currently, we identified some of the prominent parameters for the different blockchains discussed in this paper by manually parsing through the multiple documentations and configuration files [16, 18, 19, 27, 31, 33, 39, 52]. Blockchain developers must provide more intuitive documentation regarding the performance tuning of their specific blockchain implementation. Consequently, benchmarking systems could automate parameter tuning to ease the benchmarking process.

Workloads and Use Cases. Workloads are well investigated by the existing benchmarking systems, and the supported workloads for each are listed in Table 1. Diablo extracts the workload trace from five real centralized applications and designs corresponding decentralized applications (DApps) to provide a realistic blockchain-specific benchmarking scenario. Blockbench provides popular database benchmarking workloads such as YCSB and small bank, which provides a good understanding of the contrast between blockchains and databases. Blockbench also supports microbenchmarks such as IO-heavy and CPU-heavy, while HyperledgerLab provides synthetic workloads such as read-heavy, update-heavy, or skewed keys. All the existing benchmarking systems also support workloads at different transaction rates. Overall, the workloads supported by the existing benchmarking systems cover many practical and synthetic use cases, ensuring a comprehensive blockchain evaluation. There are also many other blockchain specific workloads available in the literature [10, 11, 57]. However, developing or extending a benchmarking system to include this extensive set of workloads would be advantageous. The evaluation results of the existing benchmarking systems are well explored and discussed in their corresponding papers. For example, Nasrulin et al. [58] highlight six different findings that summarize the performance of the compared blockchains. However, relating the evaluation results to the implementation specifics of the blockchains and the intended use cases would be helpful for a client trying to choose the ideal blockchain. Gramoli et al. [34] observe that Diem and Avalanche do not support challenging hardware configurations but also point out that such configurations may not be the intended use case for these blockchains. They also highlight that blockchains with eventual consistency scale better, providing a client who requires immediate consistency with realistic expectations. The intended use cases of a specific blockchain and its implementation specifics, such as its consistency and fault tolerance models, need to be effectively understood and explored while discussing benchmarking results.

Performance Metrics. The existing benchmarking studies evaluate a wide range of performance metrics. Gromit focuses on the peak transaction throughput, the maximum throughput supported by a system before it hangs. Diablo measures the average throughput and a throughput time series, including the peak throughput. BCTMark focuses more on system metrics such as CPU and memory usage. Blockbench measures the success throughput, while HyperledgerLab evaluates the committed throughput, including failed transactions. The importance and reasoning of each of the metrics are well-defined in the benchmarking studies. However, a single benchmarking system that provides a comprehensive set of all the different metrics would be valuable.

5 Related Work

The literature proposes various benchmarking systems and corresponding benchmarking studies for permissioned blockchains, which we have analyzed in our case study. Dinh et al. developed the first benchmarking system for permissioned blockchains with a precise definition for the different abstraction layers [29]. HyperledgerLab [10], which uses the Hyperledger Caliper benchmarking system [41], implemented an automated blockchain (Fabric) network deployment tool to simplify benchmarking experiments. Saingre et al. proposed a blockchain benchmarking system that adheres to the six criteria for a good benchmark [65, 67]. Nasrulin et al. investigated the popular consensus protocols and benchmarked representative blockchain systems for each [58]. Gramoli et al. implemented realistic distributed applications to evaluate multiple blockchain systems' performance uniformly [34]. The existing publications focus on developing a benchmarking system, while our work highlights general benchmarking guidelines for the blockchain community, which includes both blockchain and benchmarking system developers. Benchmarks and benchmarking systems are well-established research areas in the database community [5, 7, 35, 62]. However, despite the similarities, the implementation and application differences demand a separate discussion for benchmarking blockchains [64].

6 Conclusion

In this paper, we analyzed five permissioned blockchains to define specific problem statements regarding four main aspects of benchmarking blockchains. We provide examples from each of the chosen platforms to clarify the problem statements. Further, we discuss a general methodology to tackle each problem statement, highlighting the need for contributions from the developers and users of blockchains and benchmarking systems. We then conducted a case study of five different permissioned blockchain benchmarking systems and the affiliated benchmarking studies based on our problem statements. We emphasize the current limitations of these systems, which can help improve the state-of-the-art. Given the implementation differences between blockchains and the numerous components, configuration parameters, and metrics specific to each blockchain, one main conclusion from our work is the need for blockchain developers to actively engage in the benchmarking space. We urge blockchain developers to quantitatively identify and define system-specific factors such as the top parameters to tune, the ideal system setup for a fixed hardware configuration or cost, targeted use cases, and performance metrics that can ease the process of benchmarking blockchains.

References

1. Snowflake to avalanche : A novel metastable consensus protocol family for cryptocurrencies team rocket (2018)

2. Amsden, Z.: The libra blockchain. Diem Technical White Paper **2020** (2020)
3. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the thirteenth EuroSys conference. pp. 1–15 (2018)
4. Baudet, M., Ching, A., Chursin, A., Danezis, G., Garillot, F., Li, Z., Malkhi, D., Naor, O., Perelman, D., Sonnino, A.: State machine replication in the libra blockchain (2019)
5. Bermbach, D., Wittern, E., Tai, S.: Cloud service benchmarking. Springer (2017)
6. Blockbench. https://github.com/ooibc88/blockbench (2020), [Online; accessed 13-October-2022]
7. Brent, L., Fekete, A.: A versatile framework for painless benchmarking of database management systems. In: Chang, L., Gan, J., Cao, X. (eds.) Databases Theory and Applications. pp. 45–56. Springer International Publishing, Cham (2019)
8. Brown, R.G., Carlyle, J., Grigg, I., Hearn, M.: Corda: an introduction. R3 CEV, August **1**(15), 14 (2016)
9. Btcmark. https://gitlab.inria.fr/dsaingre/bctmark (2020), [Online; accessed 13-October-2022]
10. Chacko, J.A., Mayer, R., Jacobsen, H.A.: Why do my blockchain transactions fail? a study of hyperledger fabric. In: Proceedings of the 2021 International Conference on Management of Data. pp. 221–234. SIGMOD/PODS '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3448016.3452823, https://doi.org/10.1145/3448016.3452823
11. Chacko, J.A., Mayer, R., Jacobsen, H.A.: How to optimize my blockchain? a multi-level recommendation approach. Proc. ACM Manag. Data **1**(1) (may 2023). https://doi.org/10.1145/3588704, https://doi.org/10.1145/3588704
12. Chainhammer. https://github.com/drandreaskrueger/chainhammer (2020), [Online; accessed 13-October-2022]
13. Chung, G., Desrosiers, L., Gupta, M., Sutton, A., Venkatadri, K., Wong, O., Zugic, G.: Performance tuning and scaling enterprise blockchain applications (2019)
14. Clique proof-of-authority consensus protocol. https://eips.ethereum.org/EIPS/eip-225 (2020), [Online; accessed 13-October-2022]
15. Configure bootnodes. https://consensys.net/docs/goquorum/en/latest/configure-and-manage/configure/bootnodes/ (2020), [Online; accessed 13-October-2022]
16. Configure static nodes. https://consensys.net/docs/goquorum/en/latest/configure-and-manage/configure/static-nodes/ (2020), [Online; accessed 13-October-2022]
17. Consensus protocols. https://docs.goquorum.consensys.net/concepts/consensus (2020), [Online; accessed 13-October-2022]
18. Corda configurations. https://docs.r3.com/en/platform/corda/4.6/enterprise/node/setup/corda-configuration-fields.html (2020), [Online; accessed 13-October-2022]
19. Corda key concepts. https://docs.r3.com/en/platform/corda/5.0-dev-preview-2/introduction/key-concepts.html (2020), [Online; accessed 13-October-2022]
20. Corda notaries. https://docs.r3.com/en/platform/corda/4.8/open-source/key-concepts-notaries.html (2020), [Online; accessed 13-October-2022]
21. Corda use case directory. https://r3.com/products/use-case-directory-all/apps-list (2023), [Online; accessed 29-May-2023]
22. Crain, T., Natoli, C., Gramoli, V.: Red belly: a secure, fair and scalable open blockchain. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 466–483. IEEE (2021)

23. Diablo blockchain benchmark suite. https://diablobench.github.io/ (2020), [Online; accessed 13-October-2022]
24. Diablo blockchain benchmark suite. https://diablobench.github.io/blockchain-howto (2020), [Online; accessed 13-October-2022]
25. Diem consensus configurations. https://github.com/diem/diem/blob/latest/config/src/config/consensus_config.rs (2020), [Online; accessed 13-October-2022]
26. Diem mempool configurations. https://github.com/diem/diem/blob/latest/config/src/config/mempool_config.rs (2020), [Online; accessed 13-October-2022]
27. Diem validator nodes. https://developers.diem.com/docs/basics/basics-validator-nodes (2020), [Online; accessed 13-October-2022]
28. Diem white paper. https://developers.diem.com/docs/technical-papers/the-diem-blockchain-paper/ (2020), [Online; accessed 13-October-2022]
29. Dinh, T.T.A., Wang, J., Chen, G., Liu, R., Ooi, B.C., Tan, K.L.: Blockbench: A framework for analyzing private blockchains. In: Proceedings of the 2017 ACM international conference on management of data. pp. 1085–1100 (2017)
30. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles. pp. 51–68. SOSP '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3132747.3132757, https://doi.org/10.1145/3132747.3132757
31. Go ethereum. https://geth.ethereum.org/docs/interface/command-line-options (2020), [Online; accessed 13-October-2022]
32. Goquorum configuration file. https://consensys.net/docs/goquorum/en/latest/configure-and-manage/configure/use-configuration-file/ (2020), [Online; accessed 13-October-2022]
33. Goquorum qlight. https://consensys.net/docs/goquorum/en/latest/concepts/qlight-node/ (2020), [Online; accessed 13-October-2022]
34. Gramoli, V., Guerraoui, R., Lebedev, A., Natoli, C., Voron, G.: Diablo: A benchmark suite for blockchains. In: 18th ACM European Conference on Computer Systems (EuroSys). p. to appear (2023)
35. Gray, J.: Benchmark handbook: for database and transaction processing systems. Morgan Kaufmann Publishers Inc. (1992)
36. Greenspan, G., et al.: Multichain private blockchain-white paper. URl: http://www. multichain. com/download/MultiChain-White-Paper. pdf pp. 57–60 (2015)
37. Gromit blockchain benchmarking tool. https://github.com/grimadas/gromit (2020), [Online; accessed 13-October-2022]
38. Hearn, M., Brown, R.G.: Corda: A distributed ledger. Corda Technical White Paper **2016** (2016)
39. How fabric networks are structured. https://hyperledger-fabric.readthedocs.io/en/latest/network/network.html (2020), [Online; accessed 13-October-2022]
40. Hyperledger besu. https://www.hyperledger.org/use/besu (2020), [Online; accessed 13-October-2022]
41. https://hyperledger.github.io/caliper/ (2020), [Online; accessed 13-October-2022]
42. Hyperledger caliper benchmarks. https://github.com/hyperledger/caliper-benchmarks (2020), [Online; accessed 13-October-2022]
43. Hyperledger foundation case studies. https://www.hyperledger.org/learn/case-studies (2023), [Online; accessed 29-May-2023]
44. Hyperledgerlab ii. https://github.com/MSRG/HyperLedgerLab-2.0 (2020), [Online; accessed 13-October-2022]

45. Klenik, A., Kocsis, I.: Porting a benchmark with a classic workload to blockchain: Tpc-c on hyperledger fabric. In: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing. pp. 290–298 (2022)
46. Kokoris-Kogias, E., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., Ford, B.: Enhancing bitcoin security and performance with strong consistency via collective signing. In: Proceedings of the 25th USENIX Conference on Security Symposium. p. 279–296. SEC'16, USENIX Association, USA (2016)
47. Li, M., Wang, Y., Ma, S., Liu, C., Huo, D., Wang, Y., Xu, Z.: Auto-tuning with reinforcement learning for permissioned blockchain systems. Proc. VLDB Endow. **16**(5), 1000–1012 (february 2023). https://doi.org/10.14778/3579075.3579076, https://doi:10.14778/3579075.3579076
48. Liu, M., Yu, F.R., Teng, Y., Leung, V.C.M., Song, M.: Performance optimization for blockchain-enabled industrial internet of things (iiot) systems: A deep reinforcement learning approach. IEEE Transactions on Industrial Informatics **15**(6), 3559–3570 (2019). https://doi.org/10.1109/TII.2019.2897805
49. Lokhava, M., Losa, G., Mazières, D., Hoare, G., Barry, N., Gafni, E., Jove, J., Malinowsky, R., McCaleb, J.: Fast and secure global payments with stellar. In: Proceedings of the 27th ACM Symposium on Operating Systems Principles. pp. 80–96. SOSP '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3341301.3359636, https://doi.org/10.1145/3341301.3359636
50. Mazzoni, M., Corradi, A., Di Nicola, V.: Performance evaluation of permissioned blockchains for financial applications: The consensys quorum case study. Blockchain: Research and Applications **3**(1), 100026 (2022). https://doi.org/https://doi.org/10.1016/j.bcra.2021.100026, https://www.sciencedirect.com/science/article/pii/S209672092100021X
51. Morgan, J.: Quorum whitepaper. New York: JP Morgan Chase (2016)
52. Multichain configurations. https://www.multichain.com/developers/blockchain-parameters/ (2020), [Online; accessed 13-October-2022]
53. Multichain data streams. https://www.multichain.com/developers/data-streams/ (2020), [Online; accessed 13-October-2022]
54. Multichain performance optimization. https://www.multichain.com/developers/performance-optimization/ (2020), [Online; accessed 13-October-2022]
55. Multichain product partners. https://www.multichain.com/product-partners/ (2023), [Online; accessed 29-May-2023]
56. Multichain white paper. https://www.multichain.com/download/MultiChain-White-Paper.pdf (2020), [Online; accessed 13-October-2022]
57. Nasirifard, P., Mayer, R., Jacobsen, H.A.: Fabriccrdt: A conflict-free replicated datatypes approach to permissioned blockchains. In: Proceedings of the 20th International Middleware Conference. p. 110–122. Middleware '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3361525.3361540, https://doi.org/10.1145/3361525.3361540
58. Nasrulin, B., De Vos, M., Ishmaev, G., Pouwelse, J.: Gromit: Benchmarking the performance and scalability of blockchain systems. In: 2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS). pp. 56–63. IEEE (2022)
59. Parity: Blockchain infrastructure for the decentralised web. https://www.parity.io/ (2020), [Online; accessed 13-October-2022]
60. Quorum blockchain in action. https://consensys.net/quorum/enterprise/ (2023), [Online; accessed 29-May-2023]

61. Quorum white paper. https://github.com/ConsenSys/quorum/blob/master/docs/Quorum%20Whitepaper%20v0.2.pdf (2020), [Online; accessed 13-October-2022]
62. Raasveldt, M., Holanda, P., Gubner, T., Mühleisen, H.: Fair benchmarking considered difficult: Common pitfalls in database performance testing. In: Proceedings of the Workshop on Testing Database Systems. DBTest'18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3209950.3209955, https://doi.org/10.1145/3209950.3209955
63. Rauchs, M., Blandin, A., Bear, K., McKeon, S.B.: 2nd global enterprise blockchain benchmarking study. Available at SSRN 3461765 (2019)
64. Ruan, P., Dinh, T.T.A., Loghin, D., Zhang, M., Chen, G., Lin, Q., Ooi, B.C.: Blockchains vs. Distributed Databases: Dichotomy and Fusion. Association for Computing Machinery, New York, NY, USA (2021), https://doi.org/10.1145/3448016.3452789
65. Saingre, D., Ledoux, T., Menaud, J.M.: Bctmark: a framework for benchmarking blockchain technologies. In: 2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA). pp. 1–8. IEEE (2020)
66. Schuh, F., Larimer, D.: Bitshares 2.0: General overview (2017)
67. Smaalders, B.: Performance anti-patterns: Want your apps to run faster? here's what not to do. Queue **4**(1), 44–50 (feb 2006). https://doi.org/10.1145/1117389.1117403, https://doi.org/10.1145/1117389.1117403
68. Solana: A new architecture for a high performance blockchain v0.8.13. https://solana.com/solana-whitepaper.pdf (2020), [Online; accessed 13-October-2022]
69. Tennakoon, D., Gramoli, V.: Smart red belly blockchain: Enhanced transaction management for decentralized applications. arXiv preprint arXiv:2207.05971 (2022)
70. Thakkar, P., Nathan, S., Viswanathan, B.: Performance benchmarking and optimizing hyperledger fabric blockchain platform. In: 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). pp. 264–276 (Sep 2018). https://doi.org/10.1109/MASCOTS.2018.00034
71. Tran, N.K., Babar, M.A., Walters, A.: A framework for automating deployment and evaluation of blockchain networks. Journal of Network and Computer Applications **206**, 103460 (2022). https://doi.org/https://doi.org/10.1016/j.jnca.2022.103460, https://www.sciencedirect.com/science/article/pii/S1084804522001102
72. Types of transactions. https://developers.diem.com/docs/transactions/txns-types/ (2020), [Online; accessed 13-October-2022]
73. Valenta, M., Sandner, P.G.: Comparison of ethereum, hyperledger fabric and corda (2017)
74. Wood, D.D.: Ethereum: A secure decentralised generalised transaction ledger (2014)
75. Writing connectors. https://hyperledger.github.io/caliper/v0.5.0/writing-connectors/ (2020), [Online; accessed 13-October-2022]
76. Zhang, J., Gao, J., Wu, Z., Yan, W., Wo, Q., Li, Q., Chen, Z.: Performance analysis of the libra blockchain: An experimental study. In: 2019 2nd International Conference on Hot Information-Centric Networking (HotICN). pp. 77–83. IEEE (2019)