# Performance Evaluation on Blockchain Systems: A Case Study on Ethereum, Fabric, Sawtooth and Fisco-Bcos

Rui Wang[1,2], Kejiang Ye[1](✉), Tianhui Meng[1], and Cheng-Zhong Xu[3]

[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
{rui.wang2,kj.ye,th.meng}@siat.ac.cn
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] State Key Laboratory of IoT for Smart City, University of Macau, Zhuhai, Macao, Special Administrative Region of China
czxu@um.edu.mo

**Abstract.** Blockchain technology is currently receiving increasing attention with widely used in many fields such as finance, retail, Internet of Things, and intelligent manufacturing. Although many blockchain applications are still in the early stage, this technique is very promising and has great potential. Blockchain is considered as one of the core technologies to trigger a new round of disruptive changes after Internet. In the future, it is expected to change the development prospects of many industries. However, the current blockchain systems suffer from poor performance which affects large-scale application. In order to better understand the performance of the blockchain systems, in this paper, we analyze four mainstream blockchain systems (Ethereum, Fabric, Sawtooth and Fisco-Bcos), and then perform a performance comparison through open source blockchain benchmarking tools. After that, we propose several optimization methods and discuss the future development of blockchain technique.

**Keywords:** Blockchain · Ethereum · Fabric · Sawtooth · Fisco-Bcos

## 1 Introduction

Blockchain is essentially a distributed ledger technique. It is the core technology of Bitcoin [1] and other virtual currencies. It can record transactions between buyers and sellers and ensure that these records are verifiable and permanently stored. At present, according to different application scenarios and user needs, blockchain can be divided into three categories: public blockchain, private blockchain, and consortium blockchain.

The *public blockchain* is the most decentralized blockchain. These public blockchains, such as Bitcoin and Ethereum [2], are not controlled by third-party organizations. Everyone can access the data records on the chain, participate in transactions, and compete for the right to generate new blocks. Program developers have no right to interfere with the users, and each participant (i.e. node) can join and exit the network freely, and perform particular operations.

The *private blockchain* is completely the opposite. The write permission of the network is fully controlled by an organization or institution, and the data access is regulated by the organization. It can be understood as a weakly centralized system. Because the participating nodes is few and have strict restrictions. Compared with public blockchains, the time for private blockchains to reach consensus is relatively short, the transaction speed is faster, the efficiency is higher, and the cost is lower. This type of blockchain is more suitable for internal use by specific institutions, such as the Linux Foundation [3].

The *consortium blockchain* is a blockchain between the public and private blockchains, which can achieve "partial decentralization". Each node on the chain usually has a corresponding physical institution or organization; participants authorize to join the network and form a stakeholder alliance to jointly maintain the blockchain operation. Similar to private blockchain, consortium blockchain has the characteristics of low cost and high efficiency and is suitable for B2B transactions such as transactions and settlement between different entities.

Due to the different design, these blockchains have different application scenarios. Table 1 compares the three different blockchain systems. The public blockchain is suitable for scenario that has high requirements on credibility and security, which does not require high transaction speed. Private blockchain or consortium blockchain is more suitable for applications with high requirements on privacy protection, transaction speed and internal supervision. The consortium blockchain's transaction confirmation time and transactions per second are greatly different from the public blockchain, and the requirements for security and performance are also higher than the public blockchain.

**Table 1.** Comparison of public, private and consortium blockchain

| | Public blockchain | Private blockchain | Consortium blockchain |
|---|---|---|---|
| Participants | Free | Permissioned | Permissioned |
| Features | Completely decentralized<br>Poor performance<br>High fault tolerance | Trusted centralization<br>High performance<br>Low fault tolerance | Partially decentralized<br>Moderate performance<br>Moderate fault tolerance |
| Use cases | Cryptocurrency | Audit, Issuance | Payment, Settlement |
| Project | Bitcoin, Ethereum | ConsenSys | Hyperledger fabric |

For example, *Ethereum* is one of the most well-known public blockchains. It provides a decentralized Ethereum Virtual Machine to process peer-to-peer contracts through its dedicated cryptocurrency Ether. *Hyperledger* [4] is the representative of the consortium blockchain. As an open consortium, Hyperledger has incubated a series of business blockchain technologies, including a distributed ledger framework, a smart contract engine, a client library, a graphical interface, a utility library, and a sample application. The current blockchain system cannot solve the impossible triangle problem of "Decentralization, Scalability and Security" [27], so we need to find a balance point to take the advantages of different blockchain systems.

In order to better understand the performance of different blockchain systems, in this paper, we analyze four mainstream blockchain systems (Ethereum, Fabric, Sawtooth and Fisco-Bcos), and then perform a performance comparison through open source blockchain benchmarking tools.

The contributions are summarized as follows:

1. A detailed performance comparison of Ethereum, Fabric [5], Sawtooth [6] and Fisco-Bcos [7] is presented.
2. Major performance bottlenecks are revealed.
3. Some future optimization methods are proposed.

The rest of the paper is organized as follows: Sect. 2 introduces the architectures of different blockchains. Section 3 describes the motivation and goals of our research. Section 4 introduces the experimental method. Section 5 presents the experimental results and proposed some possible optimizations; Sect. 6 introduces the related work. Finally, we conclude the whole paper and present the future work in Sect. 7.

## 2 Background: Blockchain Architecture

### 2.1 Ethereum

The blockchain is derived from bitcoin. Generally, we call it *blockchain 1.0*, which is mainly based on various electronic currencies. The most common industry applications are micropayments, foreign exchange, and so on. With the development of blockchain, *blockchain 2.0* has emerged. The usage scenarios of Blockchain 2.0 are also richer than Blockchain 1.0. It can not only be used in payments, but can also be used in stocks, bonds, futures, loans, mortgages, property rights, smart property and smart contracts. Bitcoin is the representative of blockchain 1.0, Ethereum is the representative of blockchain 2.0. Ethereum is a platform, including digital currency Ether and Ether-Script, which are used to build distributed applications. It can implement Turing-complete virtual machines and use any currency, protocol and blockchain. The overall architecture of Ethereum can be divided into three layers [26]: *underlying services*, *core layer*, and *top-level applications* (see Fig. 1).

The *underlying services* include P2P network services, LevelDB database, cryptographic algorithms, and basic services such as sharding optimization. Each node in a P2P network is equal and provides services together. Nodes in the network can generate or review new data. The Ethereum blocks, transactions, and other data are ultimately stored in the LevelDB database. Cryptographic algorithms are used to ensure the privacy of data and the security of the blockchain. Sharding optimization makes it possible to verify transactions in parallel.

The *core layer* contains core elements such as the blockchain, consensus algorithm, and Ethereum virtual machine. It takes blockchain technology as the main body, supplements Ethereum's unique consensus algorithm, and uses EVM (Ethereum Virtual Machine) to run smart contracts. This layer is the core component of Ethereum. The first problem that the decentralized ledger of the blockchain structure needs to solve is how to ensure the consistency and correctness of the ledger data on different

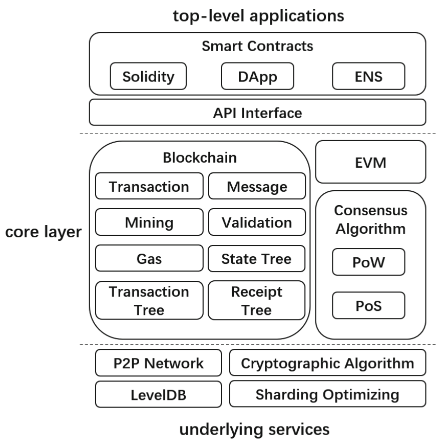

**Fig. 1.** Ethereum architecture.

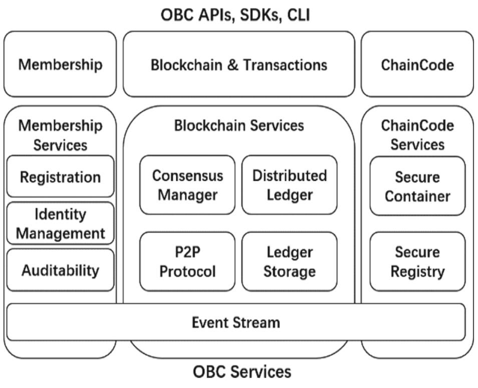

**Fig. 2.** Fabric architecture.

nodes, and the consensus algorithm is used to solve this problem. EVM is a major innovation of Ethereum. It is the operating environment of smart contracts in Ethereum, which enables Ethereum to implement more complex logics.

The *top-level applications* include API interfaces, smart contracts, and Decentralized Application (DApp). Ethereum's DApp exchanges information with the smart contract layer through Web3.j. All smart contracts run on the EVM and use RPC calls.

Various layers cooperate with each other and perform their duties to form a complete Ethereum system. In the underlying services, data such as transactions and blocks are stored in the LevelDB database. Cryptographic algorithms are used to encrypt block generation and transaction transmission. Optimization of sharding speeds up transaction verification. The consensus algorithm is used to solve the consistency of the ledger among P2P network nodes. The DApp in the top-level application needs to be executed on the EVM.

## 2.2 Hyperledger Fabric

Figure 2 shows the architecture of Fabric. Member management [23] provides member registration, identity protection, content confidentiality, and transaction auditing functions. All members of OBC (Open Blockchain) must be licensed to initiate transactions, which is different from the public blockchain (all participants do not need to log in and can submit directly). When an OBC member initiates a transaction, if the Transaction Certificate Authority (TCA) function is enabled, the transaction certificate will protect the member ID from being seen by unrelated parties. Block services [28] are used to maintain a consistent distributed ledger throughout the network. Based on the P2P communication network (gRPC), messages are transmitted between nodes through HTTP messages. Highly optimized design makes the status synchronization efficient and reliable. Consensus algorithms (PoW [8], PoS [9], PBFT [10], Raft [11]) are modular and pluggable. OBC provides a CLI client tool to enable developers to quickly test the ChainCode [22] or query the transaction status. ChainCode is used to form a smart contract and is embedded in the transaction. All confirmation nodes must execute it when confirming the transaction. ChainCode's execution environment is a sandbox (Docker [12]) and supports Go, Java, Node.js [24].

## 2.3 Hyperledger Sawtooth

Sawtooth's design includes three main architectural layers: *ledger layer*, *log layer*, and *communication layer* (see Fig. 3):

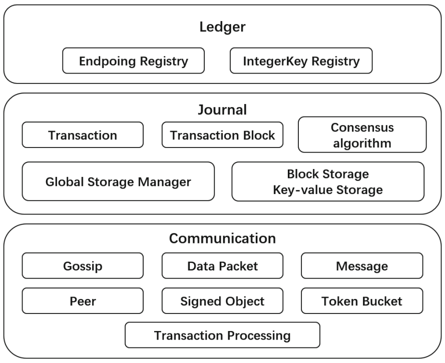

**Fig. 3.** Sawtooth architecture

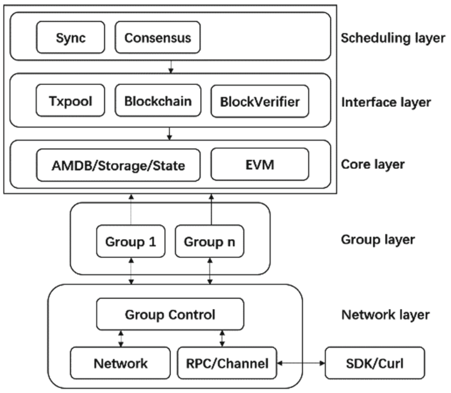

**Fig. 4.** Fisco-Bcos architecture

The implementation of the *ledger layer* is basically completed by extending the functions of the log layer and the communication layer. For example, the two built-in Endpoint Registry and IntergerKey Registry transaction families, and the MarketPlace transaction family as an example, are derived by extending the underlying functions.

The *log layer* implements the core functions of Sawtooth. It implements consensus algorithms, transactions, blocks, global storage managers, and data storage (block storage and key-value storage). The block and transaction concepts are similar with other blockchain projects.

The *communication layer* mainly implements communication between nodes through the gossip protocol [13], which mainly includes protocol layer connection management and basic flow control. Nodes send messages to each other to exchange information. Information is usually encapsulated and transmitted in different types of messages, such as transaction messages, transaction block messages, and connection messages. Like many distributed systems, in the entire architecture, lots of messages need to be sent between nodes through a chat protocol. To this end, the communication layer implements a Token Bucket mechanism to control the transmission speed of data packets.

### 2.4 Fisco-Bcos

Fisco-Bcos' structure (see Fig. 4) is mainly divided into *network layer* and *group layer*. The network layer is mainly responsible for communication between blockchain nodes. The group layer is mainly responsible for processing intragroup transactions. Each group runs a separate ledger. In a network adopting a group architecture, there may be multiple different ledgers according to different business scenarios. Blockchain nodes can select groups to join according to business relationships and participate in the data sharing and consensus process of the corresponding ledgers.

The group architecture has good scalability. Once an organization participates in such a consortium blockchain, it has an opportunity to flexibly and quickly enrich business scenarios and expand business scale, and the system operation and maintenance complexity and management costs also linearly decrease. On the other hand, each group in the group structure independently executes the consensus process, and each group independently maintains its own transaction transactions and data without being affected by other groups. The advantage is that the groups can be decoupled, operate independently, and achieving better privacy isolation. When messages are exchanged across groups, authentication information is carried, which is credible and traceable.

## 3 Motivation

Nowadays, the poor performance is one of the main challenges of current blockchain technology. The performance indicators of the blockchain mainly include *transaction throughput* and *latency*. Transaction throughput represents the number of transactions that can be processed at a fixed time, and latency represents the response and processing time to transactions. In practical applications, two factors need to be comprehensively examined. It is incorrect to consider only transaction throughput without latency. Long-term transaction response will hinder user experience and affect users' experience. Considering latency without throughput will cause lots of transactions to be queued. Some platforms must be able to handle large amount of concurrent users. Technical solutions with low transaction throughput will be directly abandoned.

In order to solve the performance problems of the blockchain systems, we have conducted in-depth research on mainstream blockchain systems, mainly including the throughput, latency, and resource utilization of the blockchain systems. By analyzing the architecture and adjusting the corresponding parameters, we understand the characteristics of each blockchain system and find out the bottlenecks of the blockchain systems. After that, our goal is to adopt some optimization measures to alleviate these bottlenecks and improve the performance of blockchain systems.

## 4 Experimental Methodology

We use *transaction throughput* and *latency* as the main performance metrics to evaluate the performance of Ethereum, Fabric, Sawtooth, and Fisco-Bcos. Transaction throughput is the number of transactions that the system can process per second.

The specific calculation method is the number of concurrent transactions divided by the average response time. Latency is the time it takes for an application to send a transaction proposal to the transaction commit. We use *caliper* to load and test the blockchain system. Caliper [14] is a blockchain performance benchmarking framework that allows users to test different blockchain solutions using predefined use cases and obtain a set of performance test results. The Caliper project was originally launched in May 2017. Huawei, a global information and communication technology company, actively participated in the design and development of the project, which was accepted by the hyperledger technical committee and added to the hyperledger project.

All tests were run in the following environments: 4 identically configured servers with the Intel (R) Xeon (R) CPU E5-2630 v4 @ 2.20 GHz CPU, 64G DDR3 RAM, 4T HDD and running Ubuntu18.04 LTS. And our test consists of three phases:

- **Preparation stage:** In this stage, the main process uses the blockchain configuration file to create and initialize internal blockchain objects, deploy smart contracts according to the information specified in the configuration, and start monitoring objects to monitor the resource consumption of the back-end blockchain system.
- **Testing phase:** In this phase, the main process performs tests based on the configuration file. Caliper will generate tasks based on the defined workload and assign them to client child processes. Finally, the performance statistics returned by each client will be stored for subsequent analysis.
- **Reporting phase:** Analyze the statistics of all clients for each test round and generate reports.

During the testing phase, we tested these blockchain systems by selecting different system settings. Each test involves sending transactions from peers at a fixed rate, and these transactions are built in a docker container. Ethereum 1.2.1, Fabric 1.4.0, Sawtooth 1.0.5, Fisco-Bcos 2.0.0 have been tested.

## 5 Experimental Results

In this section, we have studied the impact of different system architectures on the performance of different blockchain systems. The TPS (Transactions Per Second) and latency obtained in the tests are the average values obtained after multiple tests.

### 5.1 Ethereum's Performance

We use the Ethereum adapter through caliper, which includes assembling connection profiles (also known as blockchain network profiles), using adapter interfaces from user callback modules; transaction data collected by the adapter, and completing examples of connection profiles. We prepare "*open*", "*query*" and "*transfer*" workloads for Ethereum. The "open" workload includes opening accounts and testing the writing performance of the ledger. The "query" workload includes querying accounts and testing the reading performance of the ledger. The "transfer" workload includes trading between accounts and testing the transaction performance of the ledger. All chaincodes to be tested must be installed on the channel and peer. Ethereum will separately set up accounts, query accounts, and conduct transactions at the same time.

First, we set the txNumber of "open", "query", and "transfer" to 100, 200, and 100, and then we continue to increase the Send Rate for testing. The results show that when we increase the Send Rate, the throughput of the query workload increases synchronously with the Send Rate. The open workload will reach a bottleneck when throughput reaches around 15, and it cannot continue to improve. The transfer workload will reach a bottleneck when the throughput reaches around 10 and cannot be further improved (see Fig. 5). In terms of latency, the query workload does not cause any latency. For the open workload and transfer workload, as the Send Rate increases, the latency will increase slightly, but it is not obvious (see Fig. 6). We also conduct corresponding tests by increasing txNumber, but the experimental results did not change significantly.

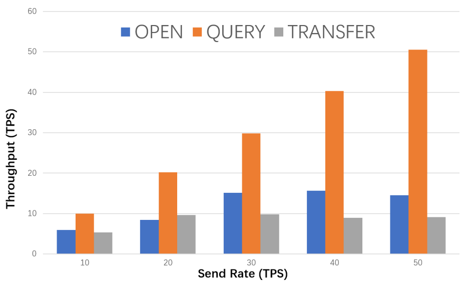

**Fig. 5.** Throughput of Ethereum with varying workload.

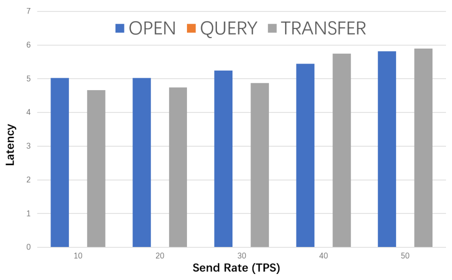

**Fig. 6.** Latency of Ethereum with varying workload.

**Discussion:** Because the block production speed of Ethereum is fixed, one block is generated every 15 s, the TPS of Ethereum is determined by the number of transactions that can be packed in a block. Ethereum has no restrictions on blocks, but the speed of network broadcasts limits the size of blocks. If the block size is too large, the latency will become very high, resulting in network unavailability. At the same time, the total amount of gas in the block will also limit the number of packaged transactions. The total amount of gas used by all transactions in the block cannot exceed this limit. Therefore, before the Istanbul upgrade, the theoretical value of TPS for Ethereum is only 30. In view of the current situation, Ethereum needs to modify the architecture in order to greatly improve the TPS. Therefore, Ethereum 2.0 (aka Serenity) is being developed. Ethereum 2.0 contains many new features: sharding, proof of stake Casper, new virtual machine eWASM, and more. These new features are currently implemented in three phases: Phase 0 mainly implements the beacon chain. The main function of the beacon chain is to implement PoS and provide the basis for sharding. In Phase 1, Ethereum 2.0 will bring a shard chain. The shard chain is the key to the future scalability of Ethereum. It allows transactions to be executed in parallel. The beacon chain will also start managing multiple shards at this time. In phase 2, various functions are beginning to be integrated, the lighthouse chain and the shard chain have been activated, and state execution will be added in this phase.

## 5.2 Hyperledger Fabric's Performance

We deploy fabric1.4.0 to 3 physical machines. Each physical machine is regarded as an Organization. Each Organization has 2 peers. Endorsement policy: Any member of Org1MSP are acceptable. The database is GolevelDB [15]. We also use "open", "query" and "transfer" workloads for Hyperledger Fabric. Then we deploy caliper on the remaining machine. We set txNumber to 200, 400, 200 respectively, and at the same time, we continuously adjusted the batchsize and Send Rate for testing. The results show that when we fixed the batchsize to 20, by increasing the Send Rate, the TPS of the "query" workload increased linearly, the TPS of the "open" workload would reach the bottleneck around 100, and the TPS of the "transfer" workload would reach the bottleneck at around 50 (see Fig. 7). In terms of latency, there is almost no latency in the "query" workload, and the latency in the "open" and "transfer" workloads will increase as the Send Rate increases (see Fig. 8).

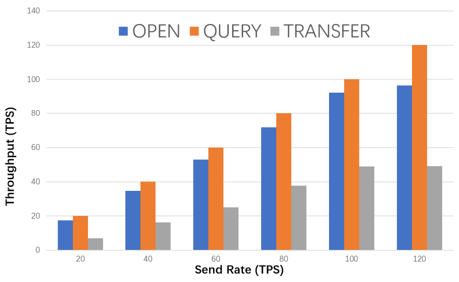

**Fig. 7.** Throughput of Fabric with varying workload (batchsize = 40).

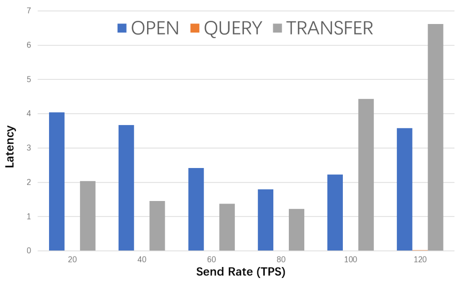

**Fig. 8.** Latency of Fabric with varying workload (batchsize = 40).

Next, we adjust the batchsize to 40, 60, 80, 100, 120, and leave the rest of the settings unchanged. The results show that with the increase of Send Rate, "transfer" workload's TPS will increase with the increase of batchsize. When batchsize is larger than 100, TPS no longer grows linearly and reaches a new bottleneck (see Fig. 9). For the latency, under the condition that the batch size is unchanged, the latency of "transfer" will decreases with the increase of Send Rate before reaching the bottleneck of TPS, and the latency will increase with the increase of Send Rate after reaching the bottleneck of TPS. As the batch size becomes larger, the latency will gradually increase. When the TPS reaches the bottleneck, the latency is gradually reduced by the effect of the batchsize (see Fig. 10).

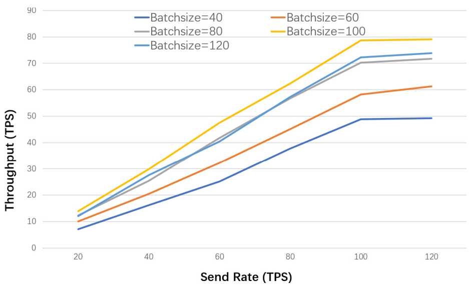

**Fig. 9.** Throughput of Fabric with different batchsize (“transfer” workload).

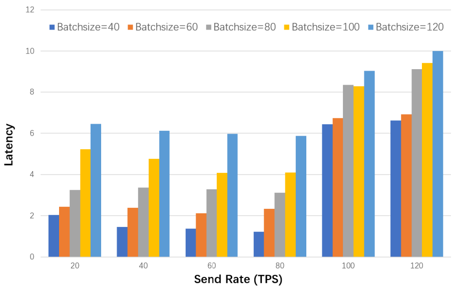

**Fig. 10.** Latency of Fabric with different batchsize (“transfer” workload).

**Discussion:** Through the experimental results, we can find that, in order to better improve the TPS and reduce the latency, when the Send Rate does not reach a threshold, we can appropriately reduce the size of the batchsize. When Send Rate exceeds the threshold, we need to choose a larger batchsize to increase the TPS and reduce the latency. Matching batchsize with Send Rate can better improve Fabric’s performance. At the same time, we noticed that the CPU resource utilization efficiency was very poor during the experiment. Therefore, improving the CPU utilization mechanism inside Fabric is also a feasible solution to improve TPS.

## 5.3 Hyperledger Sawtooth’s Performance

For Sawtooth, we first select Sawtooth 1.0.5 as the test benchmark. By modifying the protocol buffer and sawtooth-sdk version levels listed as dependencies in *packages/caliper-sawtooth/package.json* in caliper, then rebuild the Caliper project and test. We prepare “query” and “smallbank” workloads for Sawtooth. The “smallbank” workload includes transaction savings, deposit checking, send payment, write check, and amalgamate operations. We set txNumber to 500, 500, the number of accounts in smallbank to 30, the number of transactions per block to 10, and then test. The results show that with the increase of the Send Rate, the TPS of the “query” workload also increases. For the “smallbank” workload, a bottleneck occurs when the throughput reaches about 44 (see Fig. 11). In terms of latency, the latency of “query” can be ignored, and the latency of “smallbank” will continue to increase with the increase of the Send Rate. The initial period will show a linear growth trend. When the Send Rate is too high, it will show an exponential growth trend (see Fig. 12). Then we adjust the number of transactions per block.

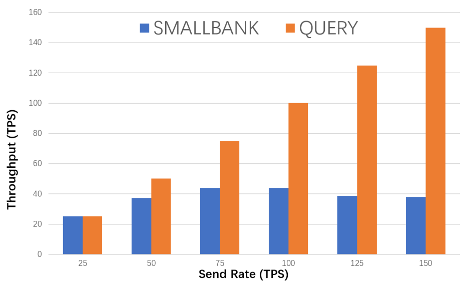

**Fig. 11.** Throughput of Sawtooth with varying workload.

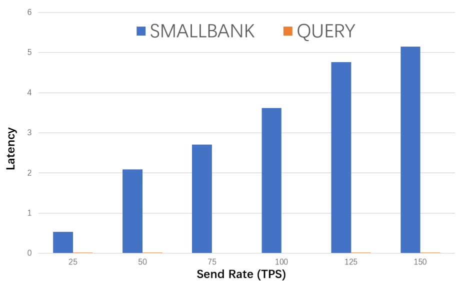

**Fig. 12.** Latency of Sawtooth with varying workload.

**Discussion:** Through testing, we found that increasing the number of transactions per block within a certain range can increase TPS. When the number of transactions per block is set to 2000, the TPS can reach about 2000. In terms of consensus algorithms, sawtooth supports a variety of consensus algorithms, such as PBFT, PoET [15], Raft, etc. With the continuous improvement of consensus algorithms, the performance of sawtooth will also be improved. In terms of performance, the development team spent a lot of energy to migrate the core components of Sawtooth from Python to Rust. As the migration work is gradually completed, the performance of sawtooth will be further improved.

### 5.4 Fisco-Bcos's Performance

For Fisco-Bcos, we first deploy our own Fisco-Bcos network. Then we add a new network configuration file and create a test script that includes initialization, run, and end phases. Finally, we add the new test script as a test round to the test profile, ensuring that the correct callback was specified for Caliper. We prepare two basic workloads "set" and "get" for Fisco-Bcos. "Set" is responsible for generating a "hello world" smart contract and deploying this smart contract. "Get" is responsible for calling the smart contract and outputting "hello world". We set txNumber to 5000, 5000, and then test. The results show that with the increase of the Send Rate, the TPS of the "get" workload increases linearly, while the TPS of the "set" workload will reach the bottleneck around 1500 (see Fig. 13). For the latency, the latency of the "set" and "get" workloads hardly changes with the Send Rate. The latency of the "set" workload is slightly higher than the "get" workload, and the "get" workload has almost no latency (see Fig. 14).

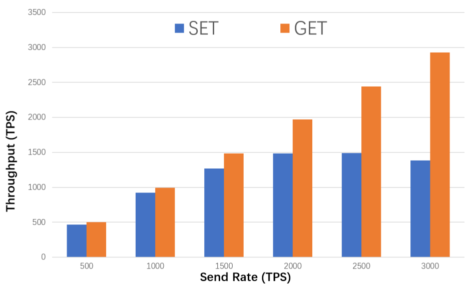

**Fig. 13.** Throughput of Fisco-Bcos with varying workload.

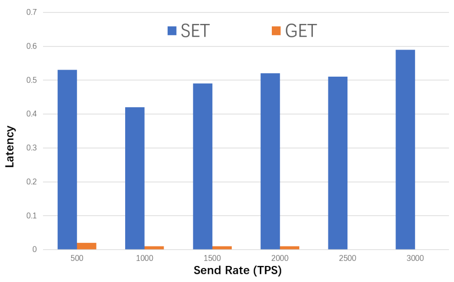

**Fig. 14.** Latency of Fisco-Bcos with varying workload.

**Discussion:** Fisco-Bcos is mainly optimized in terms of network transmission models and computing storage processes which provides great help for performance improvement. In terms of architecture, from the perspectives of storage, networking, and computing, Fisco-Bcos is upgraded around high availability and high ease-using. At the same time, based on the design principles of modularity, tiering, and pluggability, Fisco-Bcos continues to reshape the core modules to ensure the robustness of the system.

## 5.5 Comparison Analysis

By comparing the four blockchain systems, we can find that in a general setup, the TPS of Ethereum is significantly lower than the other three systems. The performance of Fabric is much better than Ethereum, but under our test conditions, it is far from the theoretical value of Fabric performance. The TPS of Sawtooth and Fisco-Bcos is better than the Fabric, which is also the consortium blockchain.

In terms of latency, the average latency of Ethereum and Fabric will be slightly larger, the average latency of Sawtooth will be smaller, and the average latency of Fisco-Bcos is the smallest (see Table 2). Due to the test platform limitation, we may not be able to measure the theoretical peak performance of the blockchain system. At the same time, because the architecture and functions of the blockchain system are not the same, we cannot use a relatively uniform workload. Therefore, the experimental results can be used as a reference.

**Table 2.** Performance comparison of 4 blockchain systems in our testing environment

| | TPS | Latency |
|---|---|---|
| Ethereum | 10–30 | 5 s |
| Fabric | 100–200 | 1–10 s |
| Sawtooth | 500–2000 | 0.5–5 s |
| Fisco-Bcos | 1500–3000 | 0.5 s |

## 6 Related Work

Due to the current performance problems of the blockchain, many systems can hardly be deployed in practice. Therefore, how to improve the performance of blockchain systems has been a popular research problem.

Dinh et al. were among the early researchers to the private blockchain. They proposed a benchmarking tool, blockbench [16], to compare the performance of Ethereum, Parity, and Hyperledger Fabric, and tested it through a set of micro and macro benchmarks. Because they studied earlier, they only studied the performance of Fabric v0.6.

Thakkar et al. conducted some research on Hyperledger Fabric v1.0, tested Fabric by adjusting configuration parameters, and proposed some simple optimization schemes based on the test results [17]. The current Fabric v1.4 architecture has many improvements compared to the old version, so many of their conclusions need to be re-examined in the new version.

Gorenflo et al. changed the fabric's architecture to reduce the calculation and I/O overhead during transaction sequencing and verification, thereby increasing the throughput from 3,000 to 20,000 [29].

Pongnumkul et al. compared the performance of Ethereum and Fabric, but the workload they choose a bit single [18]. Rouhani et al. analyzed the performance of two Ethereum clients, Geth and Parity [19]. Ampel et al. analyzed the performance of Sawtooth and identified some potential problems [20]. Hao et al. studied the impact of consensus algorithms on the performance of private blockchains [21].

Hyperledger Caliper is a blockchain performance benchmark framework, which allows users to test different blockchain solutions with predefined use cases and get a set of performance test results. This project is developing rapidly, and currently supports many projects in Hyperledger, and it is still expanding.

The development of the blockchain system is fast, and many past studies can no longer serve as a reasonable reference. At the same time, many new blockchain platforms are constantly appearing. Therefore, we need to conduct a new evaluation of the current mainstream blockchain system performance.

## 7 Conclusion and Future Work

In this work, we firstly analyzed the architecture of Ethereum, Hyperledger Fabric, Hyperledger Sawtooth and Fisco-Bcos in detail. Then we used the Hyperledger caliper as the benchmark tool and tested these blockchain systems in detail. We take transaction throughput and latency as the main performance metrics, install test tool in the blockchain systems, deploy smart contracts according to the information specified in the configuration, and start monitoring objects to monitor the resource consumption of the backend blockchain system. According to our defined workloads, the test tool will test the blockchain systems. A comprehensive analysis of the performance of the blockchain system was made by adjusting parameters such as Send Rate and batchsize, and finally the results were obtained. Based on the analysis results, we give some possible performance optimization schemes. We can see that the performance gap between the public blockchain and the consortium blockchain is very large. Therefore, Ethereum needs to develop a new generation as soon as possible to improve their performance. For Fabric, as one of the most concerned members in the consortium blockchains, its performance is not as good as the emerging consortium blockchains. Sawtooth is also an open source distributed ledger platform. It is also used to run smart contracts and aims at digital financial asset management. The overall architecture is clear and highly modular, so the ability to customize is also strong. Fisco-Bcos is derived from the Ethereum C++ version. After years of development, major changes have been made in terms of scalability, performance, and ease-using. Fisco-Bcos 2.0 has added a group architecture to overcome the bottleneck of system throughput and its performance is very good.

There are still many shortcomings in this experiment. Due to the configuration of the experimental environment, the performance we get is far from the theoretical performance. In terms of workloads, we have only a few types of workloads that make it impossible to perform a complete assessment of the performance of the entire blockchain system.

In the future, we plan to use cloud services to conduct larger-scale experiments. We will continue to study the performance optimization methods of blockchain systems, and at the same time add consensus algorithms to our research direction. In addition, we will conduct more in-depth research on the architecture of the blockchain systems to improve the performance of the blockchain systems.

**Acknowledgment.** This work is supported by Key-Area Research and Development Program of Guangdong Province (NO. 2020B010164003), National Natural Science Foundation of China (No. 61702492), Shenzhen Basic Research Program (No. JCYJ20170818153016513), Shenzhen Discipline Construction Project for Urban Computing and Data Intelligence, Science and Technology Development Fund of Macao S.A.R (FDCT) under number 0015/2019/AKP, and Youth Innovation Promotion Association CAS.

## References

1. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf. Accessed 9 Jan 2020
2. Ethereum blockchain app platform. https://www.ethereum.org/. Accessed 9 Jan 2020
3. The Linux Foundation Homepage. https://www.linuxfoundation.org/. Accessed 9 Jan 2020
4. Hyperledger Homepage. https://www.hyperledger.org/. Accessed 9 Jan 2020
5. Hyperledger Fabric Homepage. https://www.hyperledger.org/projects/fabric. Accessed 9 Jan 2020
6. Hyperledger Sawtooth Homepage. https://www.hyperledger.org/projects/sawtooth. Accessed 9 Jan 2020
7. Fisco-Bcos Homepage. http://www.fisco-bcos.org/. Accessed 9 Jan 2020
8. Jakobsson, M., Juels, A.: Proofs of work and bread pudding protocols (extended abstract). In: Preneel, B. (ed.) Secure Information Networks. ITIFIP, vol. 23, pp. 258–272. Springer, Boston, MA (1999). https://doi.org/10.1007/978-0-387-35568-9_18
9. King, S., Nadal, S.: Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. Self-published paper, vol. 19 (2012)
10. Castro, M., Liskov, B.: Practical Byzantine fault tolerance. OSDI **99**, 173–186 (1999)

11. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm. In: 2014 USENIX Annual Technical Conference (USENIX ATC 2014), pp. 305–319 (2014)
12. Docker Homepage. https://www.docker.com/. Accessed 9 Jan 2020
13. Demers, A., et al.: Epidemic algorithms for replicated database maintenance. ACM SIGOPS Oper. Syst. Rev. **22**(1), 8–32 (1988)
14. Hyperledger Caliper Homepage. https://hyperledger.github.io/caliper/. Accessed 9 Jan 2020
15. Level DB Database Homepage. https://github.com/a/leveldb. Accessed 9 Jan 2020
16. Dinh, T.T.A., et al.: Blockbench: a framework for analyzing private blockchains. In: Proceedings of the 2017 ACM International Conference on Management of Data, pp. 1085–1100. ACM (2017)
17. 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. IEEE (2018)
18. Pongnumkul, S., Siripanpornchana, C., Thajchayapong, S.: Performance analysis of private blockchain platforms in varying workloads. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN), pp. 1–6. IEEE (2017)
19. Rouhani, S., Deters, R.: Performance analysis of ethereum transactions in private blockchain. In: 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 70–74. IEEE (2017)
20. Ampel, B., Patton, M., Chen, H.: Performance Modeling of Hyperledger Sawtooth Blockchain. In: 2019 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 59–61. IEEE (2019)
21. Hao, Y., et al.: Performance analysis of consensus algorithm in private blockchain. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 280–285. IEEE (2018)
22. Chaincodes. http://hyperledger-fabric.readthedocs.io/en/release-1.1/chaincode4noah.html. Accessed 9 Jan 2020
23. Membership Service Providers (MSP). http://hyperledger-fabric.readthedocs.io/en/release-1.1/msp.html. Accessed 9 Jan 2020
24. Node SDK for Fabric Client/Application. https://github.com/hyperledger/fabric-sdk-node. Accessed 9 Jan 2020
25. Omohundro, S.: Cryptocurrencies, smart contracts, and artificial intelligence. AI Matters **1** (2), 19–21 (2014)
26. Yan, Y., Zheng, K., Guo, Z.: Ethereum Technical Details and Actual Combat, 1st edn. China Machine Press, Beijing (2018)
27. On sharding blockchains. https://github.com/ethereum/wiki/wiki/Sharding-FAQ. Accessed 9 Jan 2020
28. Barger, A., et al.: Scalable communication middleware for permissioned distributed ledgers. In: Proceedings of the 10th ACM International Systems and Storage Conference, p. 1, May 2017
29. Gorenflo, C., et al.: FastFabric: scaling hyperledger fabric to 20,000 transactions per second. In: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 455–463. IEEE (2019)
30. Chen, L., Xu, L., Shah, N., Gao, Z., Lu, Y., Shi, W.: On security analysis of proof-of-elapsed-time (PoET). In: Spirakis, P., Tsigas, P. (eds.) SSS 2017. LNCS, vol. 10616, pp. 282–297. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69084-1_19