From Use Case to Benchmark: Comparing Consortium and Private Blockchains

Maxwell Chan[a], Hengyu Luo[a], Eric Z. Ma[b] and Dah Ming Chiu[b]

[a] *The Chinese University of Hong Kong; intern at DTL* [b] *Digital Transaction Limited*

Abstract:

We discuss how consortium and private blockchains are two types of permissioned blockchains that can be used to support common applications (with slightly different semantics). By designing two benchmarks – Exchange and Loan - for these common use cases, we can compare these two types of blockchains by selecting a representative platform for each. We find that private blockchains can significantly outperform consortium blockchains, which is a useful consideration for application designers.

1. Introduction

Blockchain is widely considered as a promising technology that will disrupt many businesses, such as financial services and supply chains. The term blockchain, however, refers to a spectrum of related but different technologies. Broadly speaking, there are three types of technologies: (a) public blockchain; (b) consortium blockchain; and (c) private blockchain. The commonality across these technologies is that they all record real-world events in the form of transactions, which are in turn organized as a cryptographically chained sequence of blocks. The difference is in how the history – i.e. the chained blocks of transactions – is created and recorded.

In the case of public blockchain, the blocks are created by the public; literally anyone willing to follow the *rules of the game such as* Proof-of-Work (PoW) can participate in organizing the transactions into blocks and chaining them together. The key point is that these rules require a lot of computation by participants (wasteful of resources), and any credible public blockchain relies on having many participants, which means the latency in recording transactions and creating blocks is high, while transaction throughput is low. The transaction throughput of Bitcoin [1], the most well-known use case of public blockchain, peaks at 7 transactions per second [2].

In the case of a consortium blockchain, the blocks are created by a set of nodes (from a consortium of organizations) with permission to record history. They efficiently organize themselves to reach consensus regarding the transactions, put them into an order and into blocks, and keep multiple copies of the blockchain. There is no need for a large number of nodes to make the recording credible, but the representation from different members of the consortium is important. The performance of consortium blockchain can reach thousands of transactions per second using carefully configured high-performance computing systems such as Hyperledger Fabric [3].

For a private blockchain, a single organization is in charge of creating and managing a particular blockchain. The private blockchain approach has the important benefit of keeping data private to the managing entity, yet recorded data can still be guaranteed tamper-proof if certain hash values of the blockchain is make available for public auditing. Private blockchains can achieve significantly higher performance than consortium blockchains, due to the less limitations to the design and higher resource usage efficiency. Under many circumstances, the use of public or consortium blockchains can be replaced by one (or more) private blockchain(s), when the public or consortium scrutiny can be addressed by other means, thus allowing the application to enjoy higher performance and other benefits of using private blockchain.

Consortium blockchains and private blockchains are both referred to as permissioned blockchains, emphasizing their commonality (nodes in the system are deployed with permission), and common difference from public blockchains (in which nodes are not trusted). In this paper, we set out to discuss their differences, especially in performance, in supporting key applications. To do a credible comparison, we first find some popular use cases for permissioned blockchains, and discuss how both consortium and private blockchains could be used to support these use cases. Based on these common use cases, we create benchmarks for doing blackbox performance tests against the underlying blockchains, and report the performance comparisons.

To represent consortium blockchains, we choose Hyperledger Fabric (hereafter referred to as Fabric) which is a widely used opensource blockchain. To represent private blockchains, we choose ParallelChain [4] which is by design a private blockchain and hence a good representative[1]. It is important to point out that our study aims at comparing these two types of blockchains, rather than individual systems.

In the rest of the paper, we first briefly describe Hyperledger Fabric and ParallelChain, and the two types of blockchains represented in Section 2. We then discuss the potential applications, identify two use cases, and design corresponding benchmarks in Section 3. The comparison results are presented in Section 4. We then briefly discuss previous works and summarize our results.

2. Systems tested

Figure 1 shows the high-level architectures of Fabric and ParallelChain as three layers common to both for comparison. The first layer executes the smart contract/chaincode. The second layer coordinates the execution and provides multiversion concurrency control (MVCC). The third layer stores the blocks.

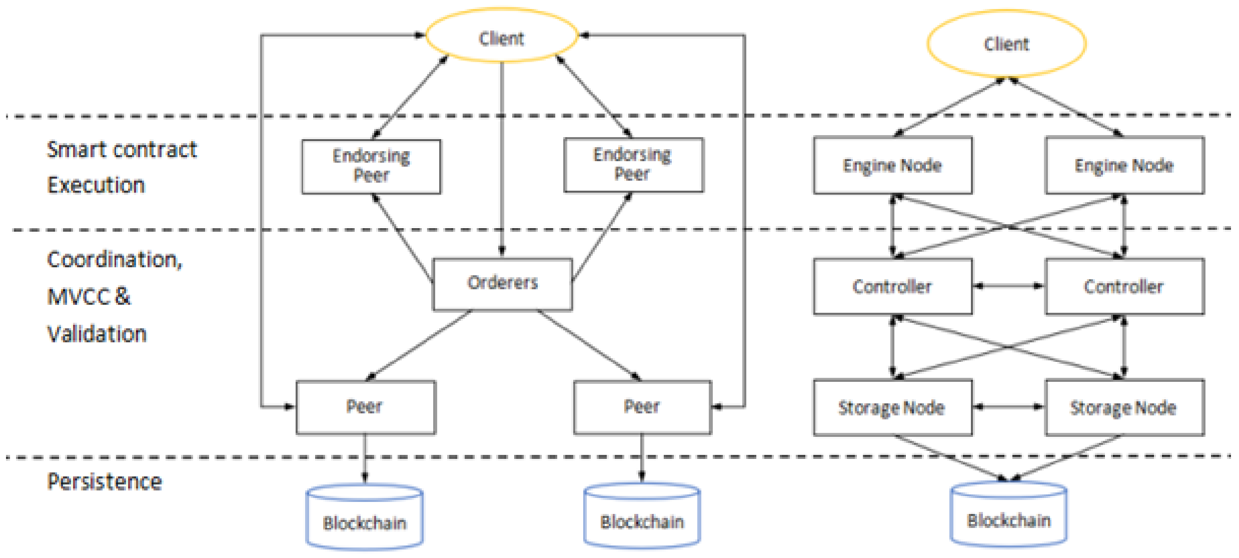

Figure 1 Basic elements of Fabric (left) and ParallelChain (right)

The differences of the architectures and workflows of the two types of blockchain systems will lead to performance differences. (1) In consortium blockchains represented by Fabric, each transaction is

[1] Quantum Ledger Database [16], a ledger-based database service offered as part of the Amazon cloud service, can be considered as another example of private blockchain as we defined.

processed at least two rounds, one by the endorsing nodes for consensus and one by all nodes for validation, whereas in a private blockchain like ParallelChain each transaction is processed only once. (2) Since each transaction is validated by all nodes in consortium blockchains, adding more nodes may not lead to higher throughput, whereas in private blockchains the nodes handle different transactions in parallel and the system is horizontally scalable. (3) In Fabric a transaction is confirmed only after the block containing it is constructed asynchronously, whereas in ParallelChain, transactions are synchronously added to the blocks and confirmed immediately. Hence, to reduce the transaction confirmation latency, the block size in Fabric is usually configured small, which affects throughput; whereas in ParallelChain, block size can be larger without affecting transaction latencies.

3. Use cases and benchmarks

3.1 Use cases

As we began to explain in the introduction, part of the excitement about blockchain is the decentralized mechanism of generating and maintaining the distributed ledger by otherwise unknown/untrusted entities, which is achieved by public blockchains. But the distributed ledger technology, when managed by permissioned nodes only, also generate a lot of interest for applying to various applications [5]. Many potential applications for permissioned blockchains are from the financial industry, which has been more conservative and has more room for applying new technology. Let us consider some examples:

a) Exchange

One example is what we call **Exchange**. There are many user accounts[2], each stores some kind of security (token), and there is a constant need to move tokens from one account to another account. One important requirement related to the general concept of an exchange is that it needs to be trusted by users, since the entity operating the exchange can easily take advantage of its users if it is not regulated or monitored/jointed operated by a consortium. There seems to be two ways to use a blockchain to deal with this problem:

a1) Use a consortium blockchain, and make each transaction endorsed by members of the consortium (could be a regulator, or multiple other operators of the exchange). The endorsing node would also keep a copy of the shared blockchain, so it can do more detailed checks offline. For ensure high transaction rate, simple online checking/consensus is assumed.

a2) Use a private blockchain, and make the history of transactions auditable by a reliable third-party. This means the history is stored in a way so it is tamper-proof (the auditor having access to certain hash values of the blockchain, and perform auditing offline.

On the assumption that both mechanisms (a1) and (a2) to ensure the integrity of the exchange are acceptable, then we can treat “exchange” as a generic use case, and create a benchmark for comparing the performance of these two solutions.

b) Loan

Another example is what we call **Loan**. In deciding to make a loan to a customer, a bank often wants to have more information than what the bank is able to collect itself. For example, is this user simultaneously applying for loans with multiple banks using the same collateral? One way to obtain such information is for banks to form a consortium to share information. Another typical scenario is trade financing, a big business for banks. Banks are actively considering the

[2] We assume all the user accounts are operated by one company.

use of blockchain to support trade financing. A slightly different scenario when multiple companies need consensus for processing transactions is the case of insurance companies processing claims; they often want to make sure that the same claim has not been paid by multiple insurance companies. In this case, the insurance companies need to check their claims history and reach consensus with each other (or simply share their claims history). The actual process for making loans or processing claims are usually very complicated, but the reason in using blockchains amounts to either for (b1) sharing transaction history, or (b2) reaching consensus for a transaction after checking against their own transaction history. Both solutions (b1) and (b2) can clearly be supported by consortium blockchains; if private blockchain is used, it is straightforward to implement (b1) and (b2) in the application layer. For this use case, we are therefore benchmarking implementing the replication and consensus process in the blockchain layer versus in the application layer.

The two use cases described above, though motivated using financial applications, capture the way blockchains are used in other scenarios as well, for example in supply chains and event tracking in various other industries. The key functions provided are (a) historical information sharing, and (b) consensus at transaction time between multiple parties.

3.2 Benchmarks

A benchmark should be considered as an abstracted but representative version of some generic application (or use case) for the technology we will test, in this case blockchain. Benchmarking methods are applied to compare performance of computers, databases, and ISP bandwidth, for example. In these cases, the benchmark is often a single program; various performance metrics are measured when this program is run on the system under test[3].

In our case, instead of creating a single program for benchmarking (which is a future goal), we create a set of basic small programs to test individual functions of the two generic use cases we identified: (a) Exchange, and (b) Loan.

The functions in the **Exchange** benchmark are summarized in the following table:

Test name	**Meaning**
Create	Create new accounts with a fixed initial amount
Delete	Delete existing account
Change	Change the amount of accounts or transform between accounts
Query	Query accounts

The functions in the **Loan** benchmark are summarized in the following table:

Test name	**Meaning**
Create User	Create unique ID for a loan borrower
Loan Request	A loan request is prepared by one bank, but need “approval” by other banks
Return Loan	An existing loan is marked as returned in current blockchain
Delete User	Delete a user

[3] An example benchmark of this type is TPC-C, used for measuring the speed for transaction processing for the well-established database technology

3.3 Implementing the benchmarks and running experiments

There are two ways to implement these benchmark functions for testing. The first method is to directly implement these functions as smart contracts, small programs that can be loaded into a blockchain platform, to emulate how the blockchain is used in a practical application. Simple test scripts can be used to launched these smart contracts in separate threads to emulate multiple concurrent users. We refer to this method as the “Direct” Method. The advantage of the **Direct** method is that it incurs very little additional system overhead. Instead of the Direct Method, we can also use systematically built benchmarking tools [6,7]. We choose Hyperledger Caliper [7], an opensource tool developed by the Hyperledger community, that can natively support different versions of Fabric as well as other blockchains by writing adaptors.

We have used both the Direct method and Caliper in our benchmarking experiments. Due to space limitation, we report only results of using Caliper. The results from the Direct Method are consistent and supportive.

3.4 Compute resources

For our experiments, we used VMs from AWS. Each blockchain node runs on a “c5.2xlarge” machine which has 8 cores and 384 IOPS capacity. Caliper is demanding in compute resources and the machines for generating workload are “c5.24xlarge” ones with 96 cores. All comparison of ParallelChain against Fabric are based on using the same compute resources.

3.5 Performance metrics

We report performance in terms of Throughput (transactions per second, or Tx/sec) and Latency (seconds, or sec). At moderate loading levels (hence little queuing delay), the product of Throughput and Latency is roughly constant, equal to the service capacity of the system.

4. Performance Results, Comparison, and Interpretations

4.1 Results for the Exchange benchmark

In Figure 9, we show the result for Fabric (HF for short), for one of the Exchange transactions, Open. The throughput saturates at around 700 Tx/sec, when the number of clients reached 5, the load level that is able to saturate the system; after that, there is a small decline. As we increase the processing nodes from 1 to 8, there is no increase in throughput. This is because virtually all nodes need to process all transactions and store a copy of the blockchain, hence no division-of-labor. The latency increases as the number of client increases, to several seconds. For the 8-node case, we notice both the Throughput and Latency curves are no longer smooth. In fact, the phenomenon of less predictable performance continues to get worse as we increase the number of nodes to be over 8. It is not unexpected because HF is designed and optimized for consortiums that are usually not very large scale.

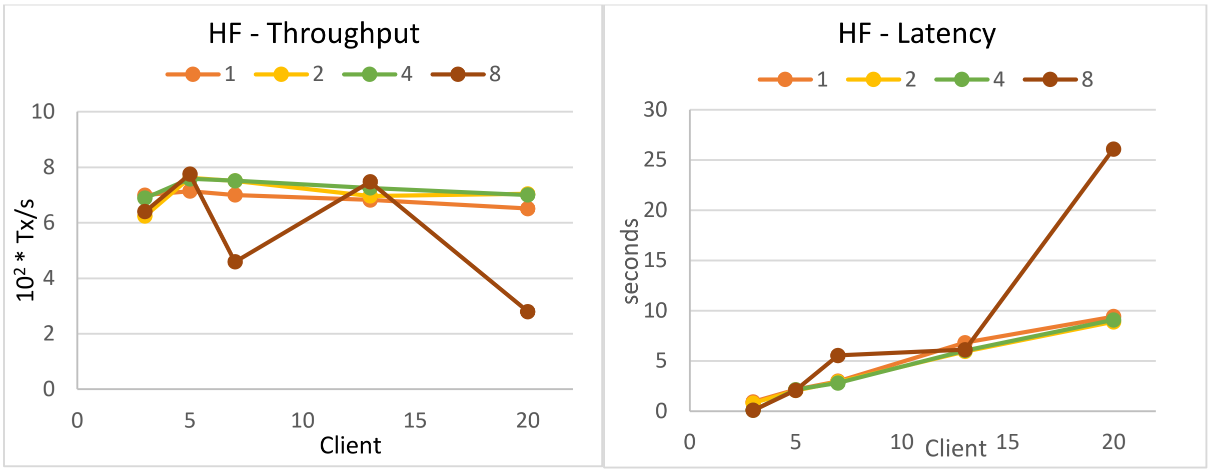

Figure 2 Throughput and Latency for HF, for different number of clients, using different number of processing nodes

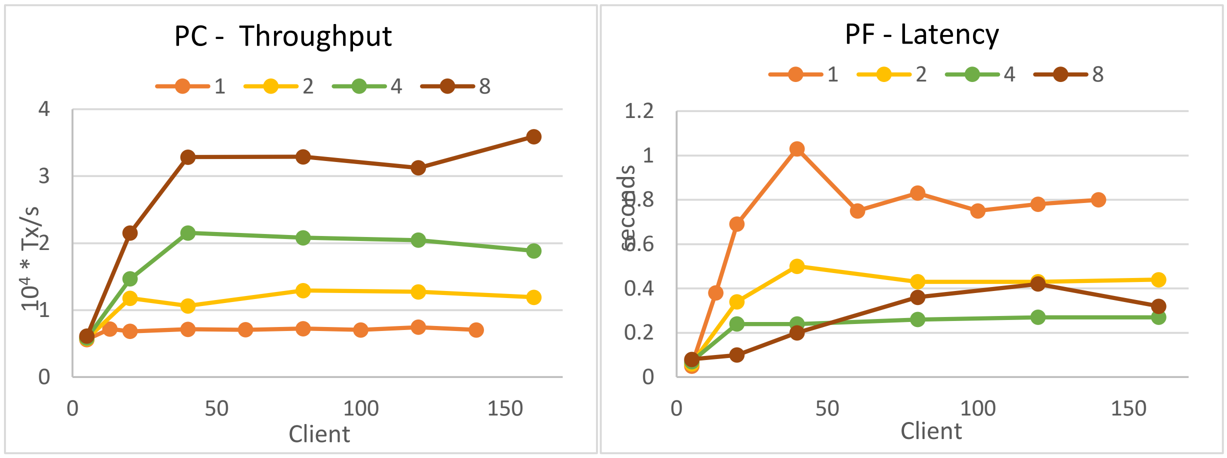

Figure 3 Throughput and Latency for PC, for different number of clients, using different number of processing nodes

This is different for PC, as shown in Figure 3. As we increase the number of processing nodes, the Throughput can reach 10s of thousands Tx/sec. Also, Latency is always kept at a fraction of a second. Performance saturates as the number of clients reaches 40 clients. As we increase the number of nodes, they can help process different transactions, hence are able to increase the throughput while keeping the latency low. As the number of clients reach around 30, the system capacity is reached, hence the Throughput saturates. The small peak of latency for the single-node case is probably because that Caliper tries to keep the total number of outstanding transactions within a limit (this is corroborated by statistics kept by Caliper).

It should be noted that even in the single-node case, when the consensus overhead for consortium blockchain is minimized, there is still a big gap between Throughput performance between PC and HF. We attribute this to the fact that each transaction needs to be processed in two phases in HF, even if there is only node, hence the higher per transaction overhead cannot be avoided.

For most of the other transactions in the Exchange benchmark, such as Change and Delete, the performance results are similar to the Open transaction. It is worth noting that the Change consecutively transaction has somewhat higher Throughput than Change randomly because accessing consecutive keys is faster; and the Change transactions are somewhat faster than Transfer transactions since the latter requires accessing two keys instead of one. But these speed reductions due to IO level overheads do not change the fact that throughput using PC is 30 to 50 times higher than using HF.

An exception is for the Query transaction, which does not involve a commit phase as it does not change the blockchain. The corresponding performance for Query for HF is only 4-5 times slower than PC because PC is configured to store one copy of the blockchain files. As a future work, we will increase replication factor and the read throughput of PC is expected to be higher.

While the Exchange benchmark is representative of use cases where all the user accounts are with a single company and other members of the consortium as just playing monitoring role, it is fair to argue that it fails to represent the situations when financial institutions (e.g. banks) need to collaborate, by sharing information when processing certain transactions. This is the reason we designed the second benchmark, Loan.

4.2 Results for the Loan benchmark

For the Loan benchmark, the transactions require consensus from all nodes. Because each node in HF validates every transaction, the Load performance on HF comes out similar to Exchange's.

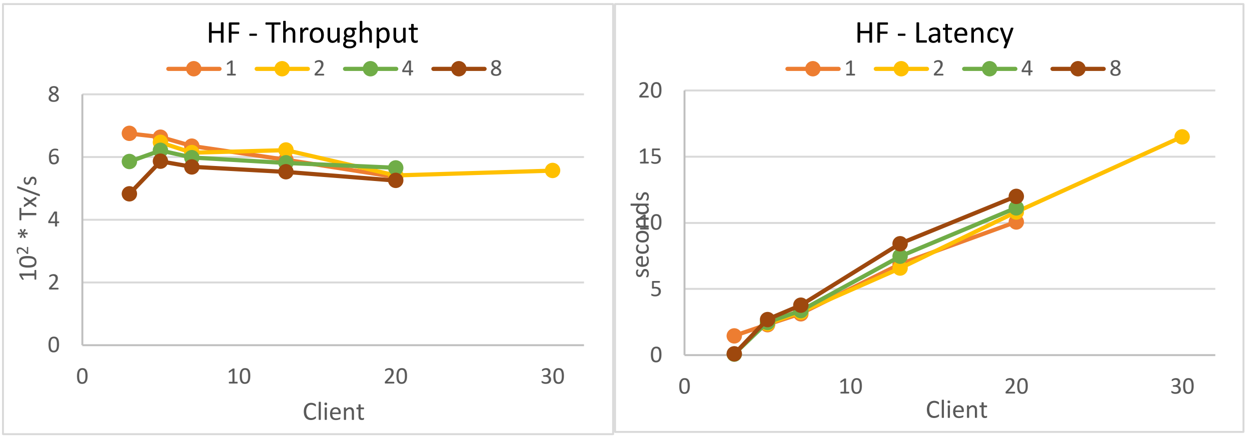

Figure 4 HF's performance applying the loan transaction in the Load benchmark

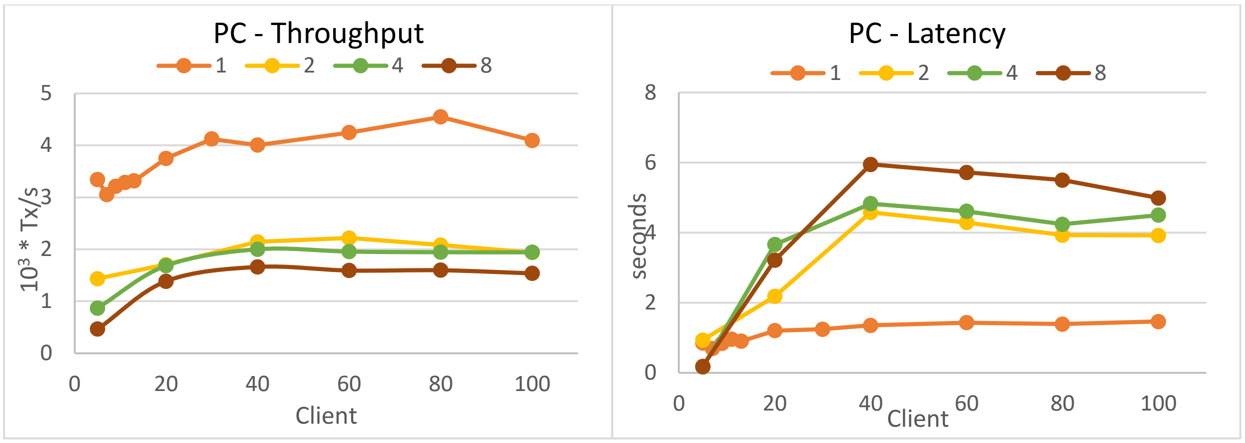

Figure 5 PC's performance applying the loan transaction in the Loan benchmark.

For PC, a two-phase-commit (2PC) step is implemented in the smart contract and application layers, and the blockchain is replicated at each node processing the transactions. This causes the throughput performance to show a similar pattern to that of HF, namely adding nodes no longer helps increase performance, but reduces performance somewhat. The case of a single node is a special case, since the 2PC is effectively avoided. But overall, the throughput for PC, at 1500-2000 Tx/sec, which is still considerably higher than HF, at around 600 Tx/sec.

The other functions in the Loan benchmark, namely Create, Repay and Delete, had similar performance (as the Loan transaction), and their charts are not reproduced here. It is worth noting that the Create function had higher performance relative to Loan, Repay and Delete for PC, because it incurs only one IO whereas the others incur at least two.

It should also be pointed out that these benchmark transactions have a higher success rate for PC than for HF. The reason is that sometimes transactions can fail due to collision in key selection. For HF, transactions simply fail; but for PC, transactions with collisions detected during 2PC layer would be retried. To completely remedy the situation, further improvement of adding retries for HF will be needed.

5 Previous Works and Discussion

There have been quite a few performance studies on permissioned blockchains published in recent years [6,8,9,10,11,12,13,14,15].

Paper [6] is a well cited performance study of permissioned blockchains. Their study includes both individual module testing as well as blackbox testing, so their study is more in depth, pinpointing the role of different system modules and their contribution to performance. They discussed how to measure performance and contributed their tool, Blockbench. Papers [8,9] study performance of permissioned blockchain from a modeling perspective. Once a model is derived, if it is accurate, it can be used to predict performance under different workload and system settings. Papers [10,11,12,13,14] focused on Fabric performance. Paper [14] is highly cited, as it also contains a very detailed discussion of the architecture and use cases for Fabric. Paper [15] studied the performance of Practical Byzantine Fault Tolerant (PBFT) systems using Fabric as an example implementation of it. PBFT is the crust of the consensus protocol adopted in consortium blockchains; understanding its performance issues helps understanding performance at the algorithm level.

In our case, we tried to point out that private blockchain is an important type of permissioned blockchains, and it can be used in many real world applications as an alternative to consortium blockchains. We explain this by identifying two common and representative use cases for permissioned blockchains, and explain how consortium and private blockchains are used. We then define benchmarks (in terms of a set of functions) for each use case, and compare these two types of blockchains on that basis. Our main conclusion, the significant performance difference in these two types of blockchains, should be useful consideration for industry practitioners. Of course, we do not claim that performance is the only consideration, and point out other considerations such as ease of auditing data immutability, whether illegitimate transactions are stopped in real time or offline, and privacy.

6 Conclusion

This article reports a systematic comparison of the performance of using consortium and private blockchains to implement a couple of common use cases of permissioned blockchains. In particular, we compared Hyperledger Fabric with ParallelChain as examples of consortium and private blockchains. Our conclusions are: (a) the use cases "exchange" and "loan" can both be support using either consortium or private blockchains, with small differences in functionality; (b) a private blockchain can be expected to outperform a consortium blockchain by a factor of 30-50 times in transaction Throughput for the Exchange benchmark, and a factor of 3-5 for the Loan benchmark. These comparisons are done using the same compute resources. The likely reasons for these performance differences have already been discussed in Section 2, when these two types of blockchains are described.

References

[1] S Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", an archived white paper, 2008

[2] K Croman et al, "On Scaling Decentralized Blockchains", 3rd Workshop on Bitcoin and Blockchain Research, 2015

[3] Hyperledger Fabric, https://cn.hyperledger.org/projects/fabric

[4] ParallelChain White Paper, https://digital-transaction.com/en_whitepaper.php

[5] K Zīle, and R Strazdiņa, "Blockchain Use Cases and Their Feasibility", Applied Computer Systems. 23(1):12-20, 2018

[6] TTA. Dinh, J Wang, G Chen, R Liu, BC Ooi and KL Tan, "Blockbench: A framework for analyzing private blockchains," Proceedings of the 2017 ACM International Conference on Management of Data, 2017

[7] Hyperledger Caliper, https://www.hyperledger.org/projects/caliper

[8] H Sukhwani, N Wang, KS Trivedi and A Rindos, "Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network)", IEEE NCA 2018

[9] L Feng, H Zhang, WT Tsai, and SM Sun, "System architecture for high-performance permissioned blockchains". Springer Frontiers of Computer Science 13(6): 1151-1165, 2019

[10] P Thakkar, S Nathan, and B Viswanathan, "Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform", IEEE MASCOTS 2018

[11] Q Nasir, I Qasse, MA Talib, and AB Nassif, "Performance Analysis of Hyperledger Fabric Platforms", Security and Communication Networks, 2018

[12] A Baliga et al, "Performance Characterization of Hyperledger Fabric", Crypto Valley Conference on Blockchain Technology, 2018

[13] S Pongnumkul, C Siripanpornchana, and S Thajchayapong, "Performance Analysis of Private Blockchain Platforms in Varying Workloads", IEEE ICCCN 2017

[14] E Androulaki et al, “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains”, EuroSys 2018
[15] H Sukhwani et al, “Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric)”, IEEE SRDS 2017
[16] Quantum Ledger Data Base (QLDB) from Amazon, https://aws.amazon.com/qldb/