Benchmark and comparison between hyperledger and MySQL

by Rz Abdul Aziz

Submission date: 13-Nov-2020 05:12PM (UTC+0700)
Submission ID: 1444854101
File name: benchmark_and_comparison_between.pdf (420.46K)
Word count: 5097
Character count: 29122

TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 18, No. 2, April 2020, pp. 705~715
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v18i2.13743

Benchmark and comparison between hyperledger and MySQL

Onno W. Purbo, Sriyanto, Suhendro, Rz Abd. Aziz, Riko Herwanto
Institut Informatika dan Bisnis Darmajaya Bandar Lampung, Indonesia

Article Info

Article history:

Received Jul 27, 2019
Revised Dec 31, 2019
Accepted Feb 5, 2020

Keywords:

Blockchain
Hyperledger
Latency
MySQL
Throughput

ABSTRACT

In this paper, we report the benchmarking results of Hyperledger, a Distributed Ledger, which is the derivation Blockchain Technology. Method to evaluate Hyperledger in a limited infrastructure is developed. Themeasured infrastructure consists of 8 nodes with a load of up to 20000 transactions/second. Hyperledger consistently runs all evaluation, namely, for 20,000 transactions, the run time 74.30s, latency 73.40ms latency, and 257 tps. The benchmarking of Hyperledger shows better than a database system in a high workload scenario. We found that the maximum size data volume in one transaction on the Hyperledger network is around ten (10) times of MySQL. Also, the time spent on processing a single transaction in the blockchain network is 80-200 times faster than MySQL. This initial analysis can provide an overview for practitioners in making decisions about the adoption of blockchain technology in their IT systems.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Riko Herwanto,
Institut Informatika dan Bisnis Darmajaya Bandar Lampung, Indonesia.
Email: rikoherwanto@darmajaya.ac.id

1. INTRODUCTION

In this work, Hyperledger Fabric [1], a Distributed Ledger Technology (DLT) [2, 3] implementation from the Linux Foundation, is benchmarked. A DLT manages Ledger through peer-to-peer networks using consensus mechanisms and smart contracts. Hyperledger is the implementation of a Blockchain framework which used to develop applications with modular architecture [4]. The Hyperledger is an open-source Blockchain and related tools project. Thus, DLT provides new models of trust and business opportunities. For this reason, DLT is a technology that emerges in many fields, such as Financial Technology (Fintech) [5], health services [6], including government organizations [7]. Unfortunately, due to complex peer-to-peer interactions, DLT performance is more difficult to access than a centralized system [8]. This work reports on an attempt to benchmark the DLT and compared it to a database system. In comparing distributed blockchain with relational databases, relational databases are more developed. Thus, there are more options in relational database frameworks than that of permissioned Blockchain frameworks [9]. The available permissioned Blockchains are all in the early development stages. It is likely more available options for permissioned Blockchains in the future. Support in cloud solutions, there is more for relational databases.

Blockchain is relatively new as compared to distributed databases. This work shows that Blockchain technology is comparable to older techniques in terms of latency [10]. In some cases, the performance is better, and when taking into account the consistency model, there will likely be several cases of meaningful use soon where the permissioned blockchain will be a better choice than the distributed database [11]. Blockchain is a new technology for sharing transactional data and calculations without using third party facilities. Blockchain uses different architecture as compared with a traditional database or protocol. The difference in architecture causes differences in performance, cost, and security, but few predict the performance of blockchain-based

systems [12]. In this paper, we have evaluated Hyperledger Fabric v1.0. The assessment shows that Hyperledger Fabric v1.0 with more than two nodes has better performance in all evaluation metrics compared to only one. Also, we are interested in comparing the performance of blockchain platforms and public blockchain with traditional databases (MySQL).

Blockchain [13], initially Block Chain, is a growing record, called blocks, which is connected and cryptographically secured. Each block usually contains a cryptographic hash from the previous block [14], timestamp, and transaction data [15]. Blockchain is designed resistant to data modification. The Blockchain is an open distributed ledger that records transactions between two parties or more efficiently and in a verifiable and permanent way [16]. As a peer-to-peer network collectively manages a distributed ledger blockchain by following specific protocols for confirming communication between nodes. Once recorded, the data in the block cannot be changed retroactively without changes to the next blocks, which requires the majority consensus of the network. The Blockchain is technically defined as a ledger to record transactions, maintained in a distributed network of non-trusting peers, in which each peer stores a copy of the ledger.

From its beginning, Blockchain is designed for safe, secure by design, and becoming an example of a distributed computing system with high byzantine fault tolerance (BFT) [17]. A decentralized consensus can be achieved by blockchain [9]. Thus, Blockchain suitable for recording events, medical records [18, 19], and other record management activities, such as identity management [20-24], transaction processing, documentation of evidence, or traceability [25]. In short, Blockchain is a ledger system that records every transaction in the form of a decentralized database network. Each data transaction is recorded in a block entity, and each block is connected (chained) to a pre-existing block. Blockchain was first mentioned by Satoshi Nakamoto when he made the world's first cryptocurrency innovation called Bitcoin in 2008. Since then, Blockchain has become the technology that supports Bitcoin's performance up to now.

One of the neat features of Blockchain is the ability to embed logical computation. When a specific criterion fulfilled, Blockchain may automatically carry out a transaction. For example, companies may program their Blockchain account to make automatic procurement raw material payments when the truck carrying material has entered the company complex. Blockchain technology has opened up opportunities for new types of applications that allow elegant data sharing across organizational boundaries where all entities can collectively own and manage shared data [26-28]. Though often confused as well as alternatives to relational databases or big data solutions, Blockchain is not a solution or a substitute for them. Blockchain is very attractive for applications that require multi-party reconciliation, trusted intermediaries, and transparency, auditable, and high-level integrity. Blockchain is publicly used in Bitcoin, Ethereum, and other cryptocurrencies as they emerge. Enterprise-scale Blockchain applications are emerging, and may soon be in production-scale deployment.

2. RESEARCH METHOD

The complex DLT architecture is divided into four layers, i.e., networks, nodes, ledgers, and application layers to facilitate further analysis. On each layer, several metrics and influence factors are determined. Different metrics and influenced by different factors are measured to benchmark. The primary and simulated DLT workload concepts are introduced. Based on this analysis, a framework was designed that built the distributed foundation from the testing environment and carried out reproducible measurements. This framework is designed so that technology is evaluated and the test environment is easily exchanged. As a result, the design framework is implemented with the benchmarking tool. For example, performance measurement and evaluation, Hyperledger Fabric, the experiments were carried out in a controlled laboratory environment. Evaluation of measurement results provides information about the performance effects of four factors, changes explicitly in transaction levels, workload, block size and also the impact of package loss. Measurements show that the factors of each layer can directly affect the performance of the entire network, increase the level of transactions, demand workloads, unfavourable memory configurations, or unfavorable network conditions. In this works, the Hyperledger blockchain platform, Hyperledger Fabric v1.0 was evaluated. The experiment was carried out on an i7 Laptop, 8GB RAM, 500GB SSD, 3 Core 2 Duo CPU, 4GB RAM, 160GB HDD, and running Ubuntu 18.04 LTE.

– **The first experiment** was to analyze the performance of the Hyperledger Fabric v1.0 platform. Platform performance evaluations assessed in terms of execution time latency and throughput by varying workloads the number of transactions and requests (requests or requests) requested simultaneously up to 20,000 transactions, the transaction is measured by the transaction submission for consensus by the partner, to add transactions to the block. Execution time is the time needed for the platform to add and execute transactions successfully. Throughput can be defined as "the number of successful transactions per second." Finally, latency in the blockchain can be measured as the time it takes for a specific platform to respond to each transaction.

- **The second Experiment** is comparing Hyperledger's work with Relational Database, in this case, a MySQL, with the same data load.
- **Throughput,** Throughput is defined as the quantity of data that is successfully transferred between nodes per unit of time, usually measured in seconds, as shown in Figure 1. Throughput is often defined for individual connections or sessions, but in some cases, the total network throughput is determined. Ideally, throughput must be the same as capacity. Capacity depends on the physical layer technology used. Network capacity must be sufficient to handle the load incurred, even when the maximum is busy in network traffic. Theoretically, throughput will increase when the load offered increases, up to a maximum of network capacity. However, network throughput depends on the access method (for example, passing token or carrier sensing), network load, and error rate. In the picture below, it shows an ideal situation, where throughput increases linearly with the load offered, and in fact, where the actual throughput decreases because the load offered reaches a certain maximum point.

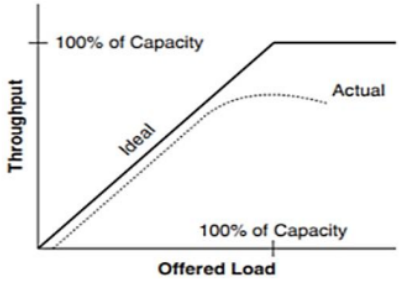

Figure 1. Throughput workflow

- **Latency,** Latency is related to the time needed to send messages from one end of the network to the other end. Latency may also be the time lag required in delivering data packets from the sender to the recipient. The higher the time lag or latency, the higher the risk of access failure. Network latency is also often interpreted as the level of delay in delivery to data communication networks and voice. Latency is measured strictly in the form of time. For example, a network to send messages takes 24 milliseconds (ms) from one end to the other. In general, there are three components of latency, namely, propagation delay, transit, and queue.

2.1. Experiment platform

In this paper, the evaluation framework for the private distributed ledger is designed and developed. For this purpose, various layers have been determined. It is the network layer, above the node layer and ledger layer to the application layer. For each of these layers, metrics, and factors have been identified, which make it possible to measure/influence the performance of DLT networks. Also, the workload has been determined, which emphasizes individual aspects of the DLT or represents realistic use cases. Four phases of the experiments are:

- Design phase

The aim is to determine the framework that runs experiments on DLT. It will support technology configuration and actual measurements and evaluation of results. The design phase is to determine various objectives, i.e., Throughput and Latency, discussed, divided into three phases, application, measurement, and evaluation, which will be discussed below.

- Testbed preparation phase

A testbed is prepared before the benchmarking process. It includes the installation and configuration of the software, i.e., OS, Docker engine, Network, necessary to facilitate the following steps. After that, Ledger Hosts, equipped with GO language and Docker CE, is needed to benchmark DLT. The testbed includes many tools, such as, Chaincode-Payload-Size, Chaincode-Scalability, Channel-Scalability, which are used during the measurement phase to record experimental hosts.

- Measurement phase

Figure 2 describes the measurement phase, starting with the initial configuration step. In running an experiment, every last configuration change is applied to the experiment Hosts, such as network interference. After this, initializing all monitors in the experiment host where the experiment takes place. It might include

recording network traffic or using resources, for example, The workload execution was initiated by Orchestration Hosts, which will not disturb the process and wait until finished. Benchmark hosts take over and operate experiments based on the definition of workload on the Ledger hosts. All previous configuration and initialization steps allow the system to run experiments without any external intervention. All changes during the experiment are timed or are directly induced by workload execution on the Ledger hosts. After the workload execution is complete, the experiment is turned off. The monitor is stopped at the Ledger host, and any interference applied during the measurement configuration step is canceled. For example, removing any artificial network connection lost that has been placed on the network, to allow a continuously not interrupted job on the testbed. Finally, each collected information is extracted from ledger hosts and collected at Orchestration Hosts. It makes it possible to immediately reset the ledger hosts except for Orchestration Hosts, for further measurements.

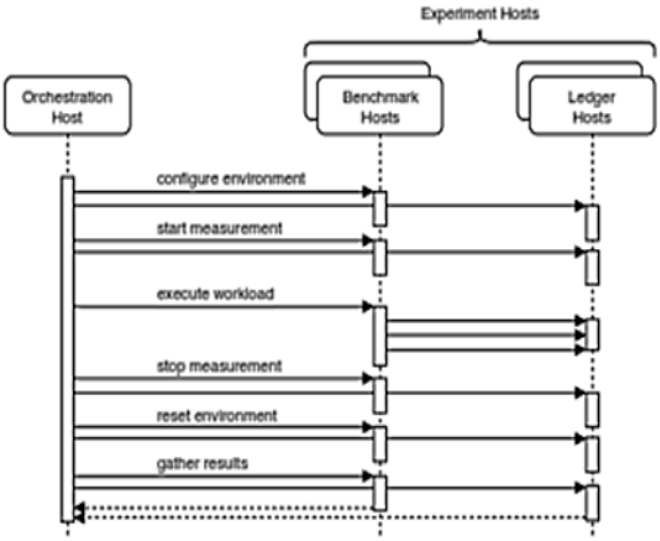

Figure 2. Measurement workflow

– Evaluation phase

The last phase is the evaluation phase, where the data collected must be evaluated, starting with the preprocessing phase. Results that come from several hosts must go through the following steps:

- Simplify further processing, such as converting to standard file formats.
- Clean any duplication of recorded traffic on many hosts.
- Normalizing time steps.
- Integration and connect various data sources.

The processed data can then be evaluated based on relevant metrics. Several methods of evaluation are defined in this work, and the format of the data that was processed beforehand allows to add further evaluation methods so that they are obtained easily:

- Transactions & Read Latency: Measure the time for transactions issued to be completed and responses available for the application that issued the transaction. Maximum, minimum, and latency for the test cycle is provided.
- Transactions & Read Throughput: Measure the flow rate of all transactions through the system, in transactions per second, during the cycle.

In the experiment, the transaction is first executed to pre-populate the chain/ledger with Block. It is the starting point for estimating how the system behaves when enough data is in the system. Then, the read-write transaction is executed where each transaction randomly reads and modifies the Block. In each experiment, one (or two) parameters vary (marked in dashed line), while keeping other parameters fixed. OS

file system cache is not removed between Insert transactions and read-write transactions assuming that, in practical settings, most of the live data will come from the file system cache. In a separate experiment, by clearing the file system cache before running the read-write experiment, the observed throughput decreased by about 17%. The experiment uses the following parameters:

- Total number of chains (default - 10)
- Number of transactions simulated in parallel on each chain (default - 10)
- Total Block in the entire chain (keys distributed evenly across the chain) (default - 20,000)
- Number of keys randomly read and modified by each transaction (default - 4)
- Size value for each Block (default - 200 bytes)
- Number of transactions in each block (default - 50)

Script Chaincode is developed to set the default experiment. The script is written using JavaScript Object Notation (JSON), which is a concise format for exchanging computer data. The format is text-based, human-readable, and is used to represent simple data structures and associative arrays (called objects). The JSON is used to send structured data through a network connection. In this work, the script to set the configuration is as follows:

```
type
type configuration struct {
    chainMgrConf *chainmgmt.ChainMgrConf
    batchConf    *chainmgmt.BatchConf
    dataConf     *dataConf
    txConf       *txConf
}

func defaultConf() *configuration {
    conf := &configuration{}
    conf.chainMgrConf = &chainmgmt.ChainMgrConf{DataDir: "/tmp/fabric/ledgerPerfTests",
NumChains: 1}
    conf.batchConf = &chainmgmt.BatchConf{BatchSize: 10, SignBlock: false}
    conf.txConf = &txConf{numTotalTxs: 20000, numParallelTxsPerChain: 2, numWritesPerTx: 4,
numReadsPerTx: 4}
    conf.dataConf = &dataConf{numKVs: 20000, kvSize: 200}
    return conf
}
```

3. RESULT AND DISCUSSION

3.1. Result performance distributed ledger technology (Hyperledger Fabric)

Shown Figure 3 the DLT increases in transaction rate up to around 257 transactions per second (tps). We simulate a network with eight (8) nodes. The distance between each node is constant at one (1) second. We generate different levels of block varying from 0 to 10 blocks. Block size sets to 50 transactions per block. The transaction arrival rate is 65 transactions per second. Throughput is directly dependent on two (2) parameters: block size, which is the number of bytes that can contain transactions in each block, and inter-block time intervals, that is, the average time needed for the system to access new block. To increase the Hyperledger throughput, one may increase the Block size and putting more transactions, or to reduce inter-block time intervals, so that the blocks are processed at a higher level. In other word, the transactions per second (tps) is affected by the increased number of nodes.

As shown in Figure 3 with increases in transactions, throughput linearly increases to around 257 tps. At that time, as shown in Figure 4 latency decreased to 100 ms with an about 62 percent decrease of latency per nodes. Latency is the time lag required in delivering data packets from the sender to the recipient. The higher the delay or latency, the higher the chance of access failure. Fortunately, latency decreases when more nodes join the transaction.

Figure 5 shows that the number of parallel transactions on the chain does not affect the overall throughput. We simulate the influence of the number of chains on the output by running the test on a different number of chains. In this experiment, we use four simulated parallel transactions on each chain, 20,000 Blocks throughout the chain with keys distributed evenly across the chain, four keys randomly read and modified by each transaction, Block size of 200 bytes, and 50 transactions in each Block. Since the process in each Block can be performed simultaneously, a process with ten (10) chains has a 25 percent higher throughput as compared with one (1) chain process. Thus, the more chains used in the process, the higher the throughput.

It is evident, considering that Chaincodes, which are included scripts for processing inter-block transactions, can be installed on nodes that link code between nodes and allow parallel chain code execution, and will affect execution time.

Figure 6 shows that increasing the number of chains increases overall throughput, the main reason seems to be the parallel validation and block commit processes. In this work, we use ten (10) chains, four (4) simulated parallel transactions in each chain, 20,000 Blocks in all chains with keys distributed evenly across chains, four (4) keys read and modified randomly by each transaction, Block size of 200 bytes, and 50 transactions in each Block.

As shown in Figure 3 increasing the number of chains will increase the throughput. As explained in Figure 3 the transactions per second (tps) increases with an increase of nodes. Since each node has a copy of the ledger, this will enable an increase in throughput. However, as shown in Figure 6 when the chain is below 100, the throughput does not appear significantly affected, around 38 percent of transactions per second. However, if one adds the number of chains up to 5 times, the percentage of throughput increases to around 62 percent.

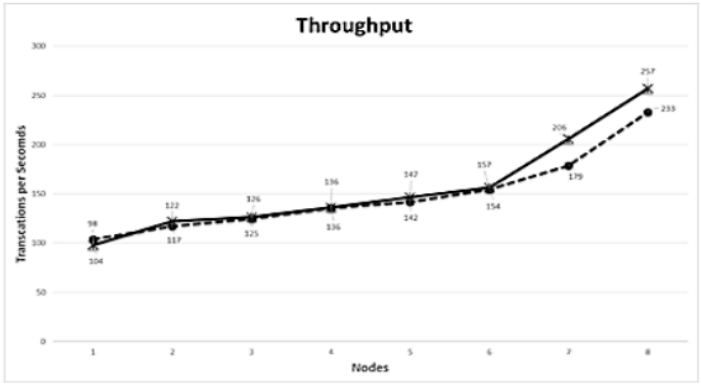

Figure 3. Throughput

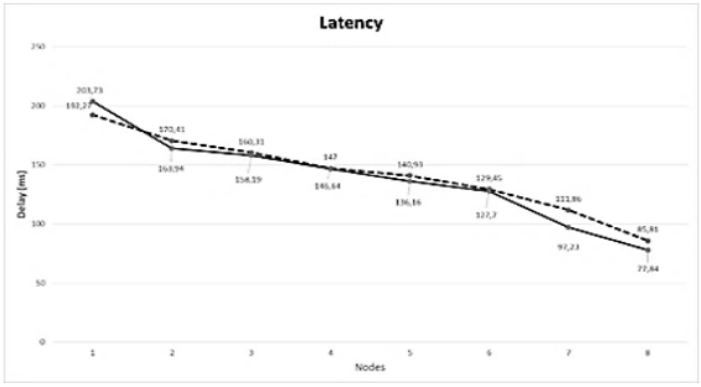

Figure 4. Latency

Figure 7 shows that increasing the number of blocks, operated in each transaction, reduces the overall throughput. In this work, we use ten chains, ten transactions simulated in parallel on each chain, 20,000 blocks in all chains with keys distributed evenly across chains, four keys read and modified randomly by each transaction, each Block sizes 200 bytes and 50 transactions in each Block which consists of a volume of transactions to be processed; any additional Block will reduce overall throughput since transactions are ordered and grouped, and sent as Block to peer. Each peer processes one block at a time.

Figure 8 shows that an increase in Block size reduces overall throughput. In this work, we use ten chains, four simulated parallel transactions on each chain, 20,000 Blocks in all chains with keys distributed evenly across chains, four (4) keys read and modified randomly by each transaction, Block size of 200 bytes, and 50 transactions in each Block. As Block size increases, the latency increases as the arrival rate increases with block size. Thus, as Block size increases, the number of transactions processed per second will rise that increase the throughput of the entire transactions.

In the experiment to generate Figure 9 we use ten (10) chains, four (4) simulated parallel transactions simulated in each chain, 20,000 Blocks in all chains with keys distributed evenly across chains, four (4) keys read and modified randomly by each transaction, Block size is 200 bytes, with 50 transactions in each Block. An increase in the number of transaction sizes will increase overall throughput. This increase has the potential because of writing simultaneous mass transactions to each Block. The increase seems to occur at more than 100 transactions.

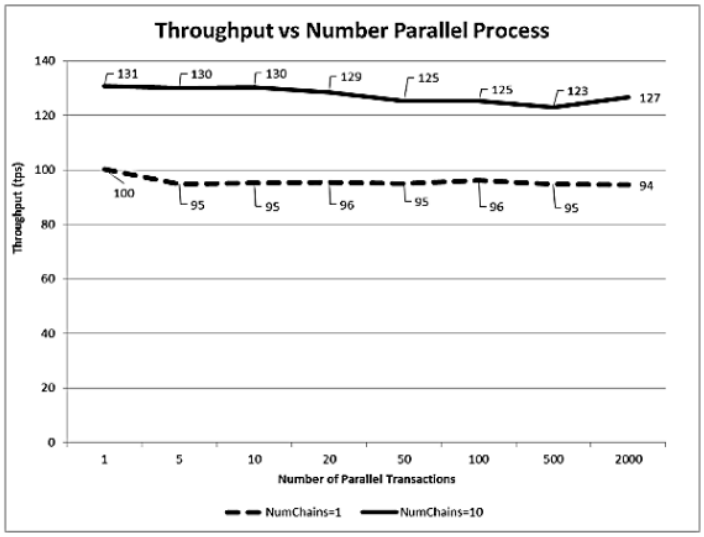

Figure 5. Throughput vs number parallel process

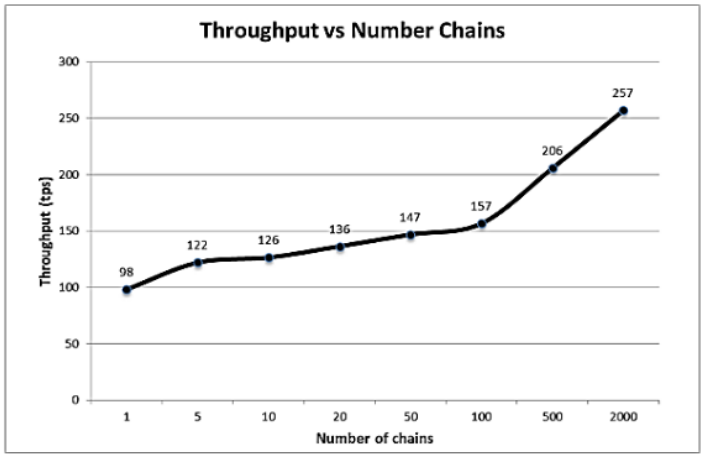

Figure 6. Throughput vs number chains

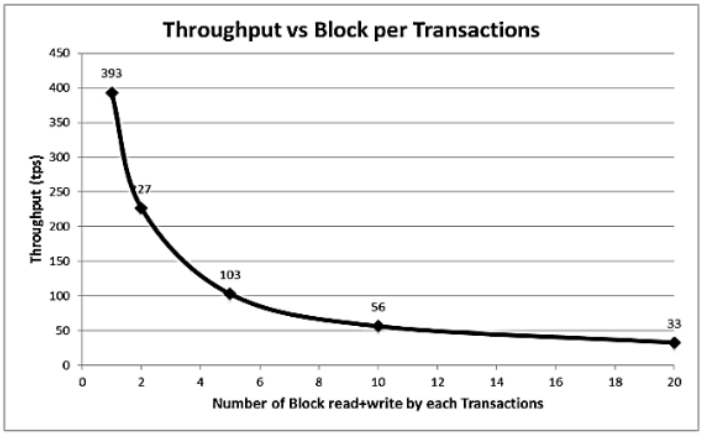

Figure 7. Throughput vs number block per transaction

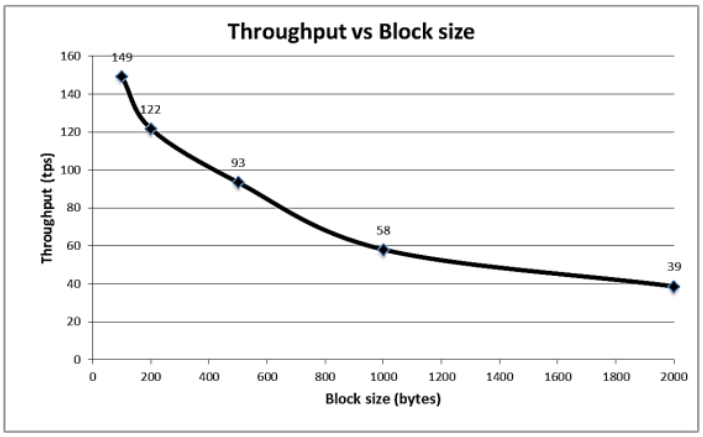

Figure 8. Throughput vs block size

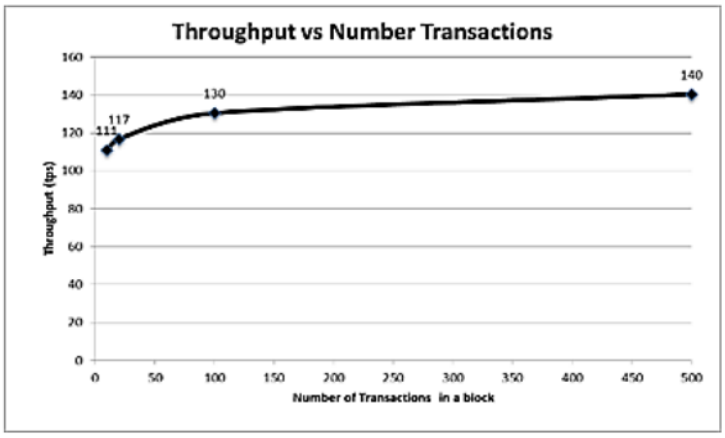

Figure 9. Throughput vs number of transaction 9

3.2. Comparing hyperledger and a relational database

In this section, HyperLedger and relational databases, such as MySQL, with two ways, read and write data will be compared. This comparison will provide some practical insights to combine blockchain technology and centralized databases. In this work, Hyperledger Fabric for the blockchain and MySQL database for relational databases are used. As for such arrangements, we have two standards. The first is whether the object tested is functionally complete or not. Second, Fabric Hyperledger to implement various functions and executions [16]. Also, MySQL can be used to store various types of data.

Figure 10 shows the execution time of each transaction, the Blockchain spends more time on a larger volume of data, reaching 2027 ms with 6KB data, the average reading and writing time is 1.22 ms. However, we found that the time used to read and write data that is spent by the Blockchain is 1660 times higher than MySQL. Figure 11 shows the relationship between the throughput and volume of data per transaction between Blockchain and MySQL database. As data volumes increase and exceed the 2KB threshold, the Hyperledger throughput increasing.

Comparing the result, Hyperledger shows higher performance with a much larger data volume and linear relationships with data. It is found that Hyperledger is 80-200 faster than MySQL. With the increase in the data volume in one transaction, the needed processing time increases. The time and volume of data used results in an exponential relationship. Note that there are eight nodes in the blockchain, a linear relationship may be the result of the number of nodes.

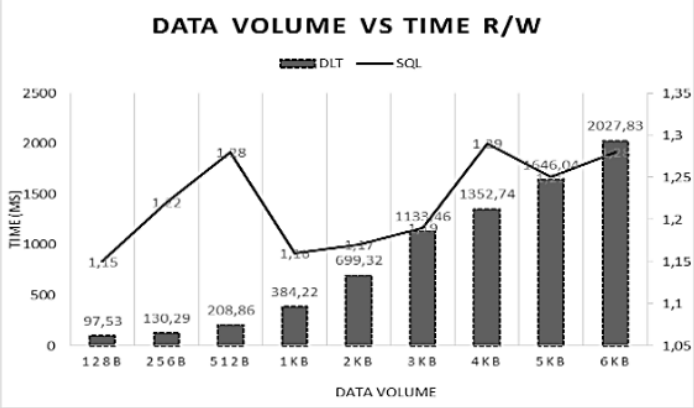

Figure 10. Data volume vs time R/W

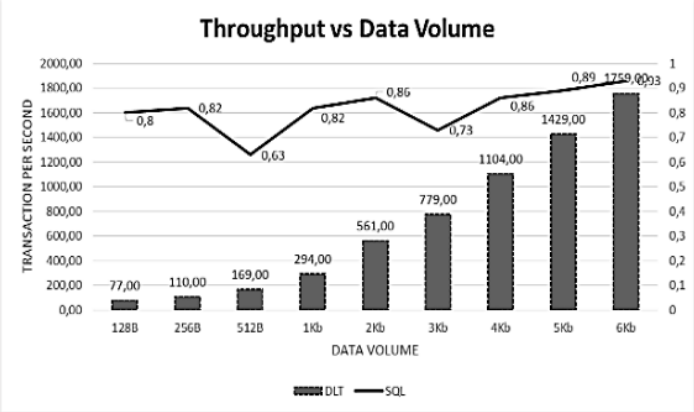

Figure 11. Throughput vs data volume

3.3. Findings

From the test results, it was found that the maximum data volume in one transaction on the Hyperledger network is around 10 times from MySQL. Also, the time spent processing one transaction on the blockchain network is 80-200 times faster than MySQL. Figure 3 shows that the level of DLT transactions increased to around 257 transactions per second. By reducing the load in the form of a chain of code causing delays in stacking, the load on the node or network is generated accumulated from the previous transaction. Figure 4 shows a slight increase in transaction latency based on workload, with a load of 10 KB. Workload shows an earlier and more significant increase in transaction latency. Also, this shows that the transaction performance increases when the block size increased. The reduction from 10 transactions per block to 5 transactions leads to doubling the average transaction latency at the high transaction level. Increased transaction rates, heavy workloads, unconfirmed block sizes, and high network losses have been confirmed to limit the performance of distributed ledgers. Also, this work shows that performance depends on parameters/conditions on different layers. Some of these parameters can be modified to improve performance. Adjusting block sizes to individual requirements from general ledger applications may improve the performance, while other settings may not be easily controlled, such as workloads or network characteristics run by distributed ledgers.

4. CONCLUSION AND FUTURE WORK

In this paper, we report a structured experimental approach to characterize performance Blockchain Hyperledger platform. The framework developed in this paper was built to be expanded. It includes adding factors, metrics, workloads, and more factors that can be extended with various network limitations such as limited network speed or network delays. Metrics must include information on the burden of each node, to enable making appropriate statements on the congestion of the ledger distributed. In this work, this feature has been partially implemented. Thus, the amount of workload may be easily increased. We found that the read throughput of the system is found to be linear while the Write process is almost linear at low transaction level below 1000 transactions per second. The read and write latency is affected by the number of nodes.

The number of participating nodes may be increased and may result in better throughput and latency. An increase in participating nodes may require infrastructure scaling to achieve the desired performance. More infrastructure scaling experiments have to be carried out on the testbed. Evaluate the limit of the distributed ledgers setup may also of interest. The ledger parameters adjustment to optimize its performance in a constraint running environment. The performance, i.e., throughput, execution time, and latency, indicates that Hyperledger is consistently better than SQL. We found that the maximum data volume in one transaction on the Hyperledger network is around ten (10) times of MySQL. Also, the time spent processing one transaction on the blockchain network is 80-200 times faster than MySQL.

In general, Hyperledger produces the same performance for a certain number of nodes, regardless of the load. However, Hyperledger performance is affected as the number of nodes is changed, and, thus, the number of Blocks, Block size, and the Number of Transactions. To achieve higher throughput, greater efficiency, the block interval should be made as little as possible. We have found that the block interval for Hyperledger based Blockchain protocol should not be less than 12s. It would ensure faster propagation and low latency. This result implies that the blockchain may be more suitable for data-intensive applications/systems.

REFERENCES

[1] E. Androulaki, *et al.*, "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains," *Proc. of the Thirteenth EuroSys Conf.*, pp. 1-15, 2018.
[2] H. Sukhwani, N. Wang, K. S. Trivedi and A. Rindos, "Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network)," *IEEE 17th Int. Symp. on Network Comp. and App,* pp. 1-8, 2018.
[3] Walport M, "Distributed Ledger Technology: Beyond Block Chain," UK Government Office for Science, Tech. Rep; 2016.
[4] S. Davidson, P. D. Filippi, and J. Potts., "Disrupting Governance: The New Institutional Economics of Distributed Ledger Technology, *SSRN Electronic Journal,* pp. 1-27, 2016.
[5] S. Chishti, and J, Barberis, "The FinTech Book: The Financial Technology Handbook for Investors, Entrepreneurs and Visionaries," *John Wiley & Sons,* 2016.
[6] J. Cunningham, J. Ainsworth, "Enabling Patient Control of Personal Electronic Health Records Through Distributed Ledger Technology," *Stud Health Technol Inform,* vol. 245, pp. 45-48, 2018.
[7] D. Genkin, D. Papadopoulos, and C. Papamanthou, "Privacy in Decentralized Cryptocurrencies," *Communications of the ACM,* vol. 61, no. 6, pp. 78-88, 2018.
[8] I. Kocsis, *et al.*, "Towards Performance Modeling of Hyperledger Fabric," *International IBM Cloud Academy Conf.* 2017.

[9] D. Chays and Y. Deng, "Demonstration of AGENDA Tool Set for Testing Relational Database Applications," *Proc. of 25th Int. Conf. on Software Engineering*, pp. 802-803, 2003.
[10] R. M. R. Yasaweerasinghelage, M. Staples and I. Weber. "Predicting Latency of Blockchain-Based Systems Using Architectural Modelling and Simulation," *IEEE Int. Conf. on Software Architecture,* pp. 253-256, 2017.
[11] Rauchs M, *et al.,* "Distributed Ledger Technology Systems: A Conceptual Framework," *SSRN Electronic Journal,* 2018.
[12] Q. Nasir, I. A. Qasse, M. A. Talib, and A. B. Nassif, "Performance Analysis of Hyperledger Fabric Platforms," *Security and Communication Networks,* vol. 2018, pp. 1-15, 2018.
[13] A. O'dowd., "Hands-On Blockchain with Hyperledger: Building Decentralized Applications with Hyperledger Fabric and Composer," *Packt Publishing,* 2018.
[14] A. Baliga, "Performance Characterization of Hyperledger Fabric," *Crypto Valley Conference on Blockchain Technology,* pp. 65-74, 2018.
[15] B. White, *et al.,* "An Integrated Experimental Environment for Distributed Systems and Networks," *Proc. of the 5th symposium on Operating systems design and implementation. Boston, Massachusetts,* vol. 36, pp. 255-270, 2002.
[16] E. Buchman, "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains," *Department Engineering Systems and Computing University of Guelph,* 2016.
[17] V. Dhillon, D. Metcalf, and M. Hooper, "The Hyperledger Project," *In Blockchain Enabled Applications, Apress, Berkeley,* pp. 139-149, 2017.
[18] S. Nakamoto, "Bitcoin: A Peer-to-peer Electronic Cash System," *BITCOIN,* pp. 1-9. 2008.
[19] C. Cachin, "Architecture of the Hyperledger Blockchain Fabric," *In Workshop on Distributed Cryptocurrencies and consensus ledgers,* vol. 310, pp. 4-11, 2016.
[20] C. Decker and R. Wattenhofer, "Information Propagation in the Bitcoin Network," *13-th IEEE International Conference on Peer-to-Peer Computing,* pp. 1-10, 2013.
[21] J. Cunningham, and J. Ainsworth. "Enabling Patient Control of Personal Electronic Health Records Through Distributed Ledger Technology," *Stud Health Technol Inform,* vol. 245, pp. 45-8, 2018.
[22] R. Bolze, *et al.,* "Grid'5000: A Large Scale and Highly Reconfigurable Experimental Grid Testbed," *The International Journal of High Performance Computing Applications,* vol. 4, pp. 481-494, 2006.
[23] K. Croman, *et al.,* "On Scaling Decentralized Blockchains," *In International Conference on Financial Cryptography and Data Security*, pp. 106-125, 2016.
[24] R. Maull, *et al.,* "Distributed Ledger Technology: Applications and Implications," *Strategic Change,* vol. 26, no. 5, pp. 481-489, 2017.
[25] N. Papadis, *et al.,* "Stochastic Models and Wide-Area Network Measurements for Blockchain Design and Analysis," *IEEE Conf. on Comp. Comm,* pp. 2546-2554, 2018.
[26] P. Sajana, M. Sindhu, and M. Sethumadhavan, "On Blockchain Application: Hyperledger Fabric and Ethereum," *International Journal of Pure and Applied Mathematics,* vol. 118, no. 18, pp. 2965-2970, 2018.
[27] S. Pongnumkul, C. Siripanpornchana and S. Thajchayapong, "Performance Analysis of Private Blockchain Platforms in Varying Workloads," *26th Int. Conf. on Comp. Commu. and Net,* pp. 1-6, 2017.
[28] P. Thakkar, "Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform," *IEEE 26th Int. Symp. on Modeling, Analysis, and Simulation of Comp and Telecomm, Sys.,* pp. 264-276, 2018.

Benchmark and comparison between hyperledger and MySQL

ORIGINALITY REPORT

10%	0%	10%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

4%

★ "Distributed, Ambient and Pervasive Interactions: Understanding Humans", Springer Science and Business Media LLC, 2018

Publication

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		